US005689565A

United States Patent [19]

Spies et al.

[11] Patent Number: 5,689,565

[45] Date of Patent: Nov. 18, 1997

[54] **CRYPTOGRAPHY SYSTEM AND METHOD FOR PROVIDING CRYPTOGRAPHIC SERVICES FOR A COMPUTER APPLICATION**

[75] Inventors: **Terrence R. Spies**, Redmond; **Jeffrey F. Spelman**, Duvall; **Daniel R. Simon**, Redmond, all of Wash.

[73] Assignee: **Microsoft Corporation**, Redmond, Wash.

[21] Appl. No.: **496,801**

[22] Filed: **Jun. 29, 1995**

[51] **Int. Cl.**[6] **H04L 9/00**

[52] **U.S. Cl.** **380/25**; 380/24

[58] **Field of Search** 380/24, 25

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,423,287	12/1983	Zeidler	380/24
4,458,109	7/1984	Mueller-Schloer	380/25
4,484,025	11/1984	Ostermann et al.	380/21
4,488,801	12/1984	Foster et al.	380/21
5,005,200	4/1991	Fischer	380/25
5,220,501	6/1993	Lawlor et al.	364/408
5,231,666	7/1993	Matyas	380/25
5,245,656	9/1993	Loeb et al.	380/25
5,323,464	6/1994	Elawder et al.	380/24
5,440,635	8/1995	Bellovin et al.	380/25
5,455,407	10/1995	Rosen	364/408
5,509,071	4/1996	Petrie et al.	380/21
5,511,121	4/1996	Yacobi	380/24
5,524,073	6/1996	Stambler	380/24
5,535,276	7/1996	Ganesan	380/25
5,557,518	9/1996	Rosen	380/24
5,559,887	9/1996	Davis et al.	380/24

FOREIGN PATENT DOCUMENTS

WO 93/10503	5/1993	WIPO	G06F 15/30

OTHER PUBLICATIONS

Products and Services Description from Commercial Announcement Document for Open Market Inc. (May 1995).

Description of Internet payment system from CyberCash, Inc. (Feb. 1995).

Description of Internet market system from Virtual Holdings, Inc. Copyright 1994–95.

Bellare et al., *IKP–A Family of Secure Electronic Payment Protocols*, Apr. 16, 1995.

Gifford et al., *Payment Switches for Open Networks* (1995).

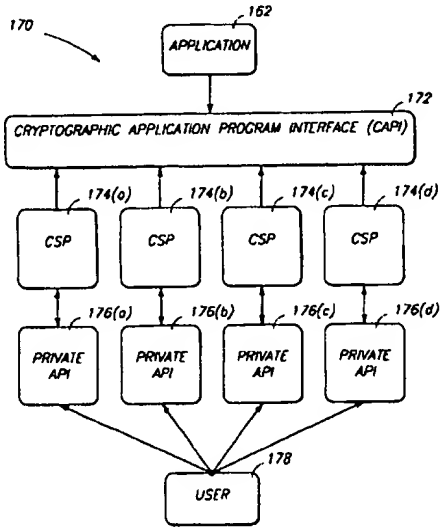
Primary Examiner—Salvatore Cangialosi

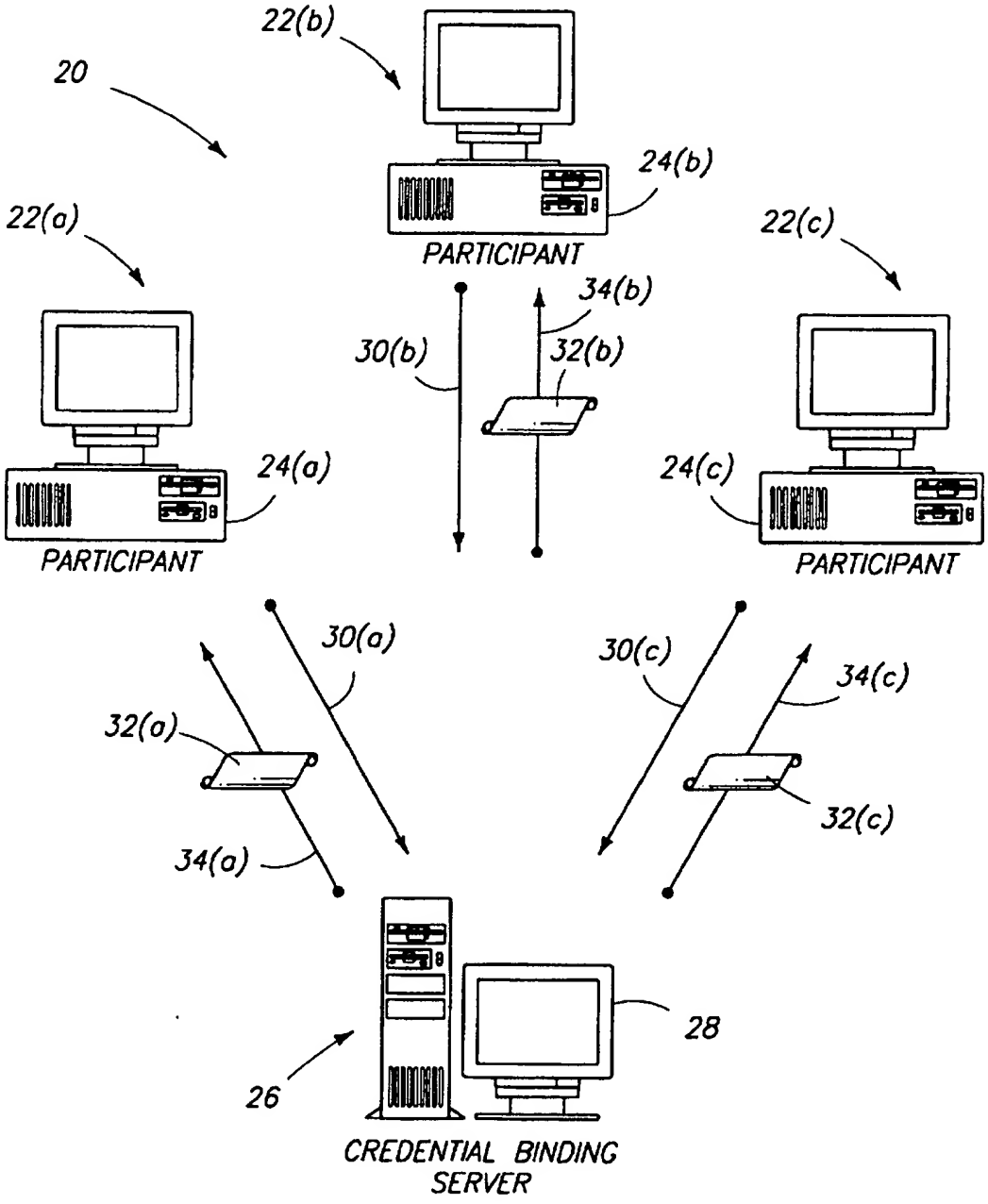
Attorney, Agent, or Firm—Lee & Hayes, PLLC

[57] **ABSTRACT**

A cryptography system architecture provides cryptographic functionality to support an application requiring encryption, decryption, signing, and verification of electronic messages. The cryptography system has a cryptographic application program interface (CAPI) which interfaces with the application to receive requests for cryptographic functions. The cryptographic system further includes at least one cryptography service provider (CSP) that is independent from, but dynamically accessible by, the CAPI. The CSP provides the cryptographic functionality and manages the secret cryptographic keys. In particular, the CSP prevents exposure of the encryption keys in a non-encrypted form to the CAPI or application. The cryptographic system also has a private application program interface (PAPI) to provide direct access between the CSP and the user. The PAPI enables the user to confirm or reject certain requested cryptographic functions, such as digitally signing the messages or exportation of keys.

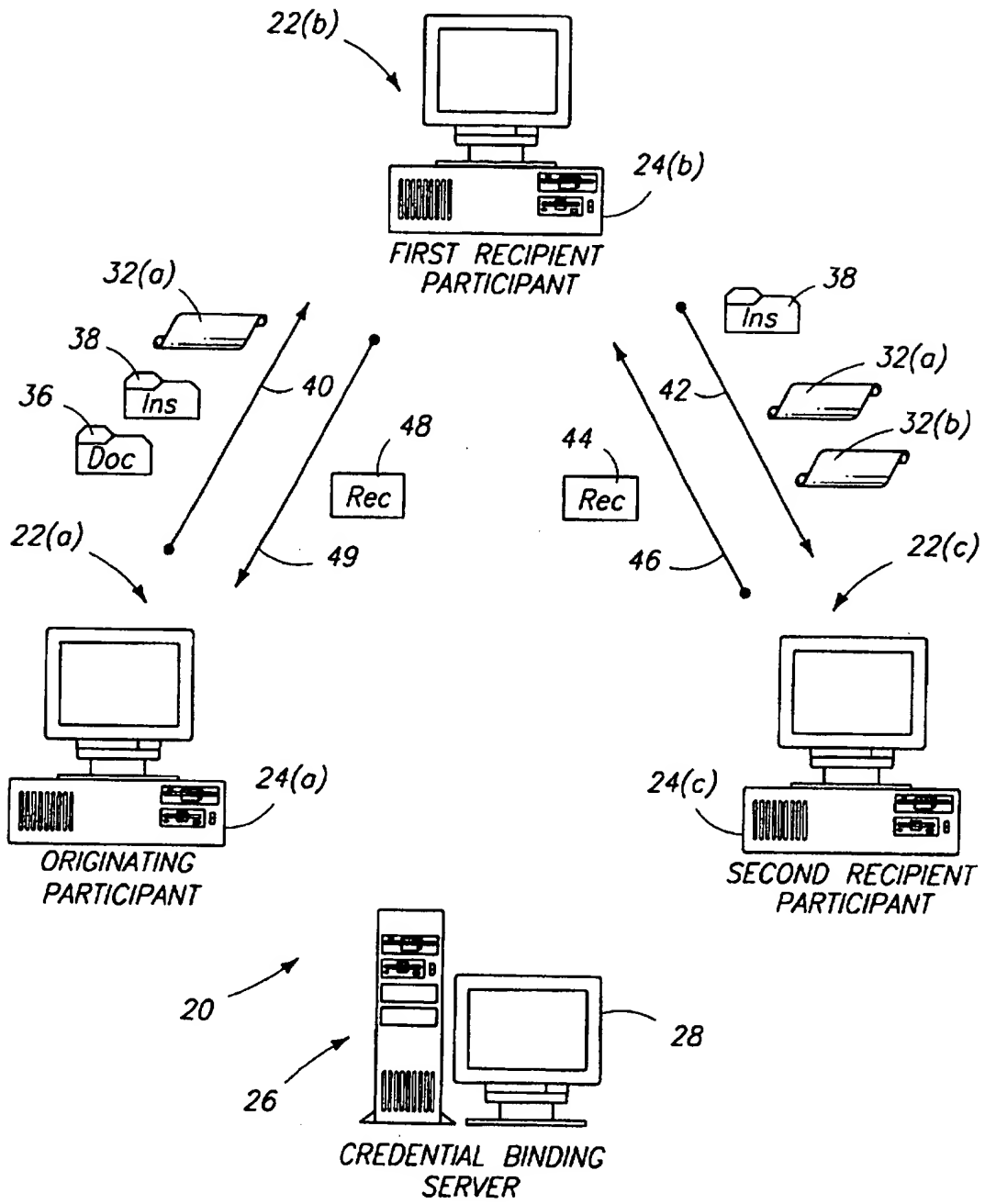
49 Claims, 23 Drawing Sheets





REGISTRATION PROCESS

Fig 1



TRANSACTION PROCESS

Fig 2

REGISTRATION PROCESS

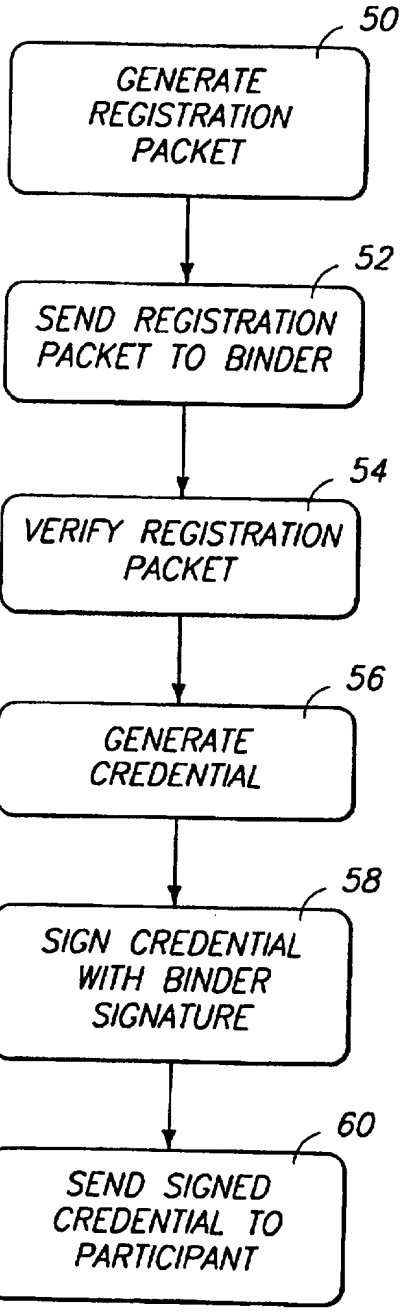


Fig 3

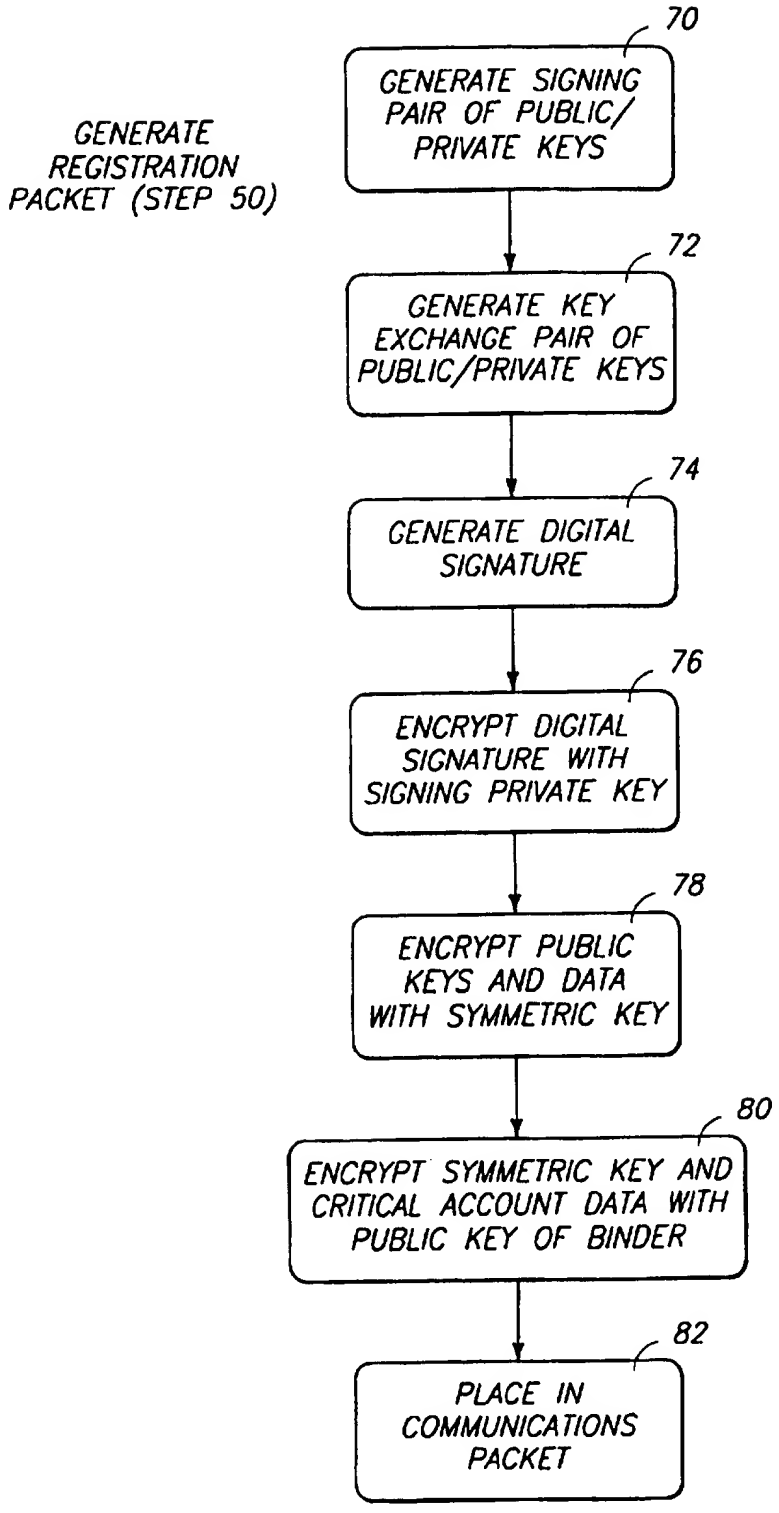

GENERATE REGISTRATION PACKET (STEP 50)
70
GENERATE SIGNING PAIR OF PUBLIC/ PRIVATE KEYS
72
GENERATE KEY EXCHANGE PAIR OF PUBLIC/PRIVATE KEYS
74
GENERATE DIGITAL SIGNATURE
76
ENCRYPT DIGITAL SIGNATURE WITH SIGNING PRIVATE KEY
78
ENCRYPT PUBLIC KEYS AND DATA WITH SYMMETRIC KEY
80
ENCRYPT SYMMETRIC KEY AND CRITICAL ACCOUNT DATA WITH PUBLIC KEY OF BINDER
82
PLACE IN COMMUNICATIONS PACKET

Fig 4

VERIFY REGISTRATION PACKET (STEP 54)

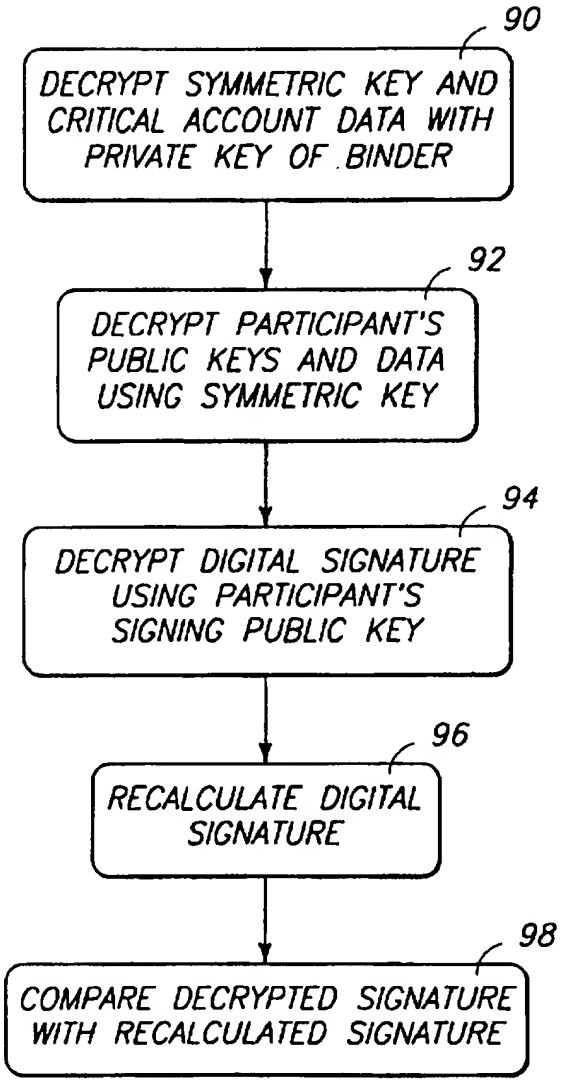


Fig 5

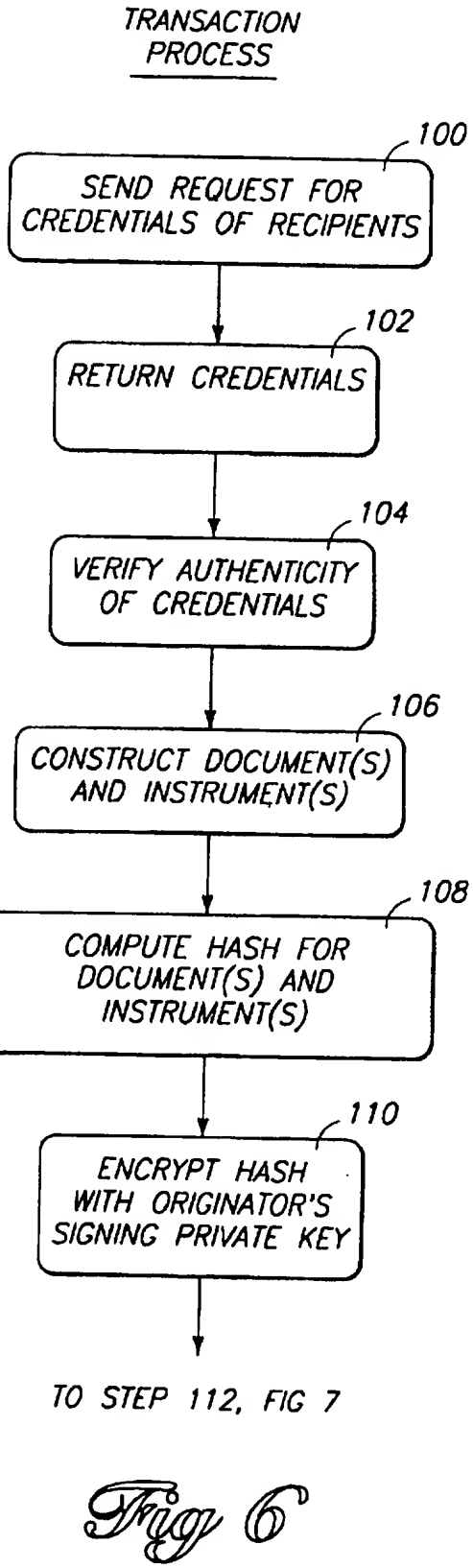

TRANSACTION PROCESS
100
SEND REQUEST FOR CREDENTIALS OF RECIPIENTS
102
RETURN CREDENTIALS
104
VERIFY AUTHENTICITY OF CREDENTIALS
106
CONSTRUCT DOCUMENT(S) AND INSTRUMENT(S)
108
COMPUTE HASH FOR DOCUMENT(S) AND INSTRUMENT(S)
110
ENCRYPT HASH WITH ORIGINATOR'S SIGNING PRIVATE KEY
TO STEP 112, FIG 7

Fig 6

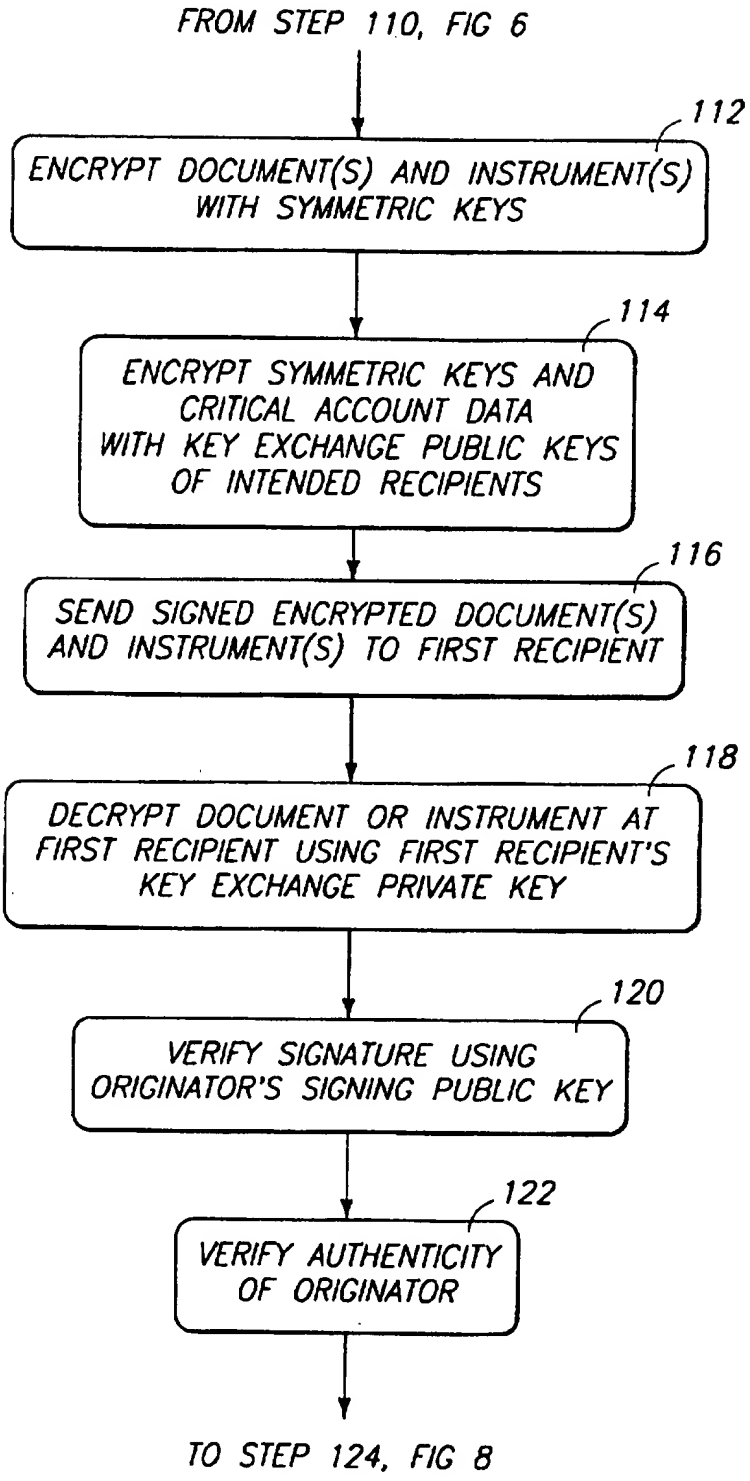

FROM STEP 110, FIG 6
112
ENCRYPT DOCUMENT(S) AND INSTRUMENT(S) WITH SYMMETRIC KEYS
114
ENCRYPT SYMMETRIC KEYS AND CRITICAL ACCOUNT DATA WITH KEY EXCHANGE PUBLIC KEYS OF INTENDED RECIPIENTS
116
SEND SIGNED ENCRYPTED DOCUMENT(S) AND INSTRUMENT(S) TO FIRST RECIPIENT
118
DECRYPT DOCUMENT OR INSTRUMENT AT FIRST RECIPIENT USING FIRST RECIPIENT'S KEY EXCHANGE PRIVATE KEY
120
VERIFY SIGNATURE USING ORIGINATOR'S SIGNING PUBLIC KEY
122
VERIFY AUTHENTICITY OF ORIGINATOR
TO STEP 124, FIG 8

Fig 7

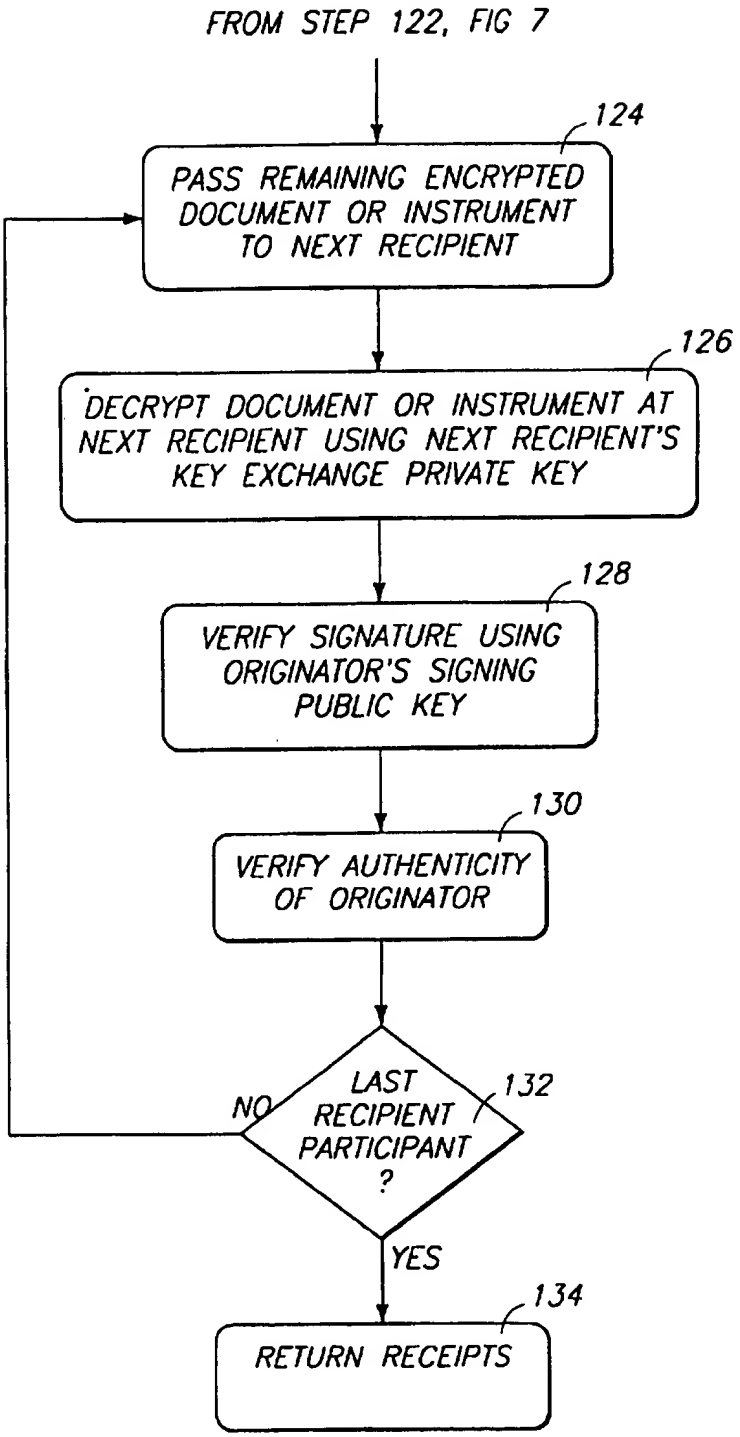

FROM STEP 122, FIG 7
124
PASS REMAINING ENCRYPTED DOCUMENT OR INSTRUMENT TO NEXT RECIPIENT
126
DECRYPT DOCUMENT OR INSTRUMENT AT NEXT RECIPIENT USING NEXT RECIPIENT'S KEY EXCHANGE PRIVATE KEY
128
VERIFY SIGNATURE USING ORIGINATOR'S SIGNING PUBLIC KEY
130
VERIFY AUTHENTICITY OF ORIGINATOR
132
LAST RECIPIENT PARTICIPANT ?
NO
YES
134
RETURN RECEIPTS

Fig 8

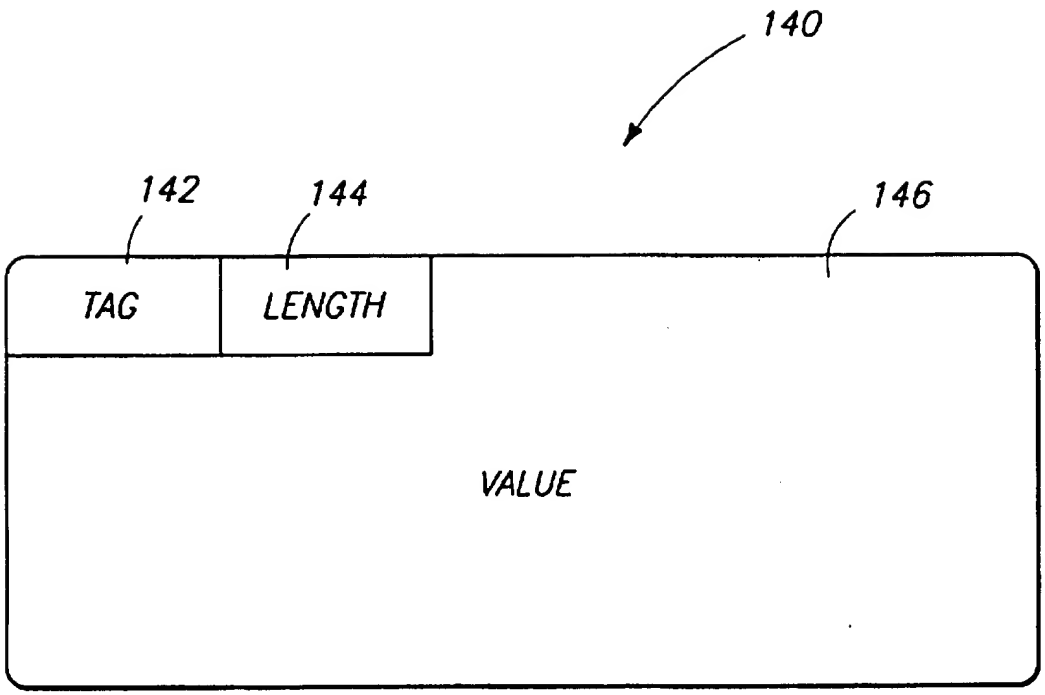


Fig. 9

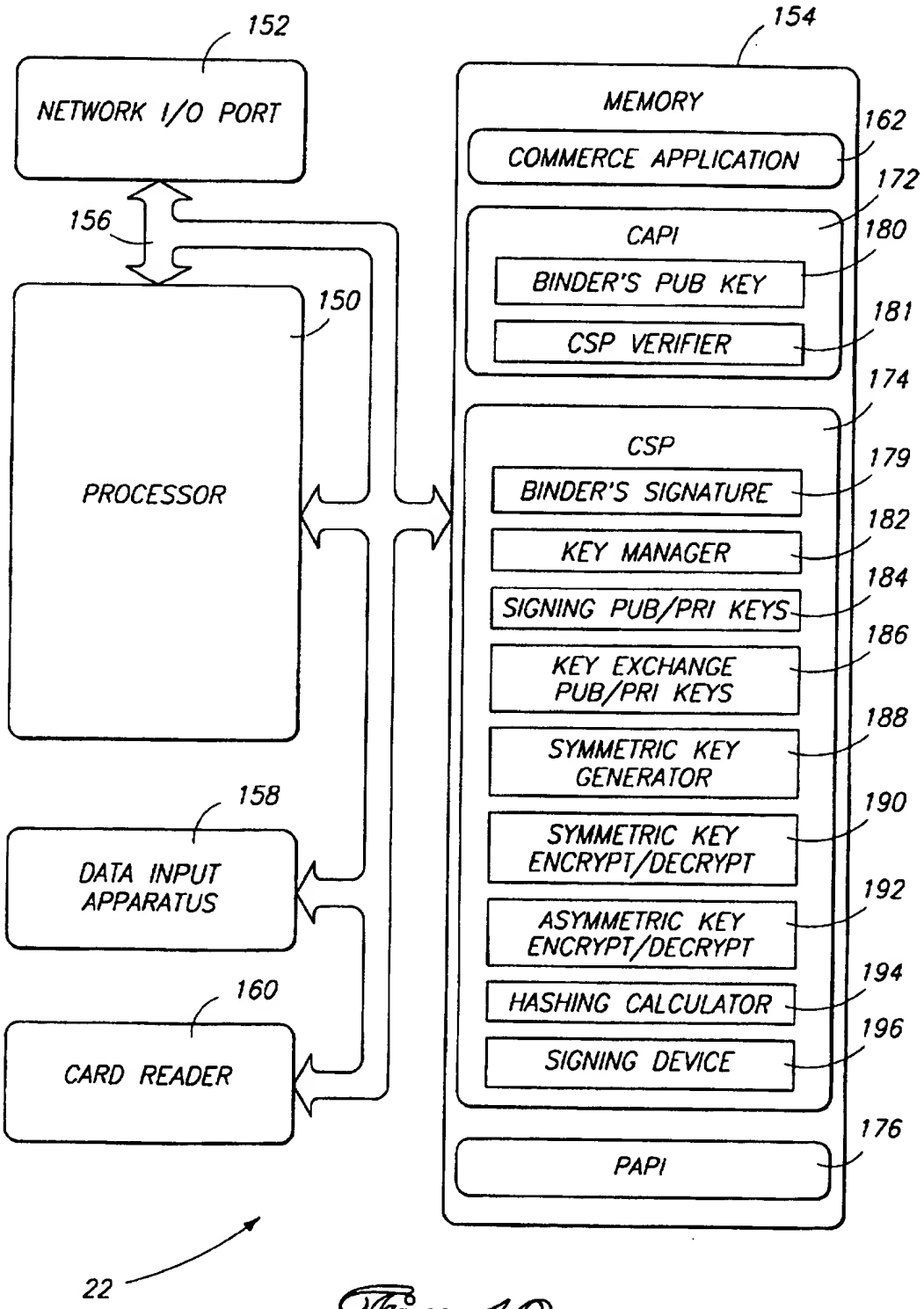

152
NETWORK I/O PORT
156
150
PROCESSOR
158
DATA INPUT APPARATUS
160
CARD READER
22
154
MEMORY
162
COMMERCE APPLICATION
172
CAPI
180
BINDER'S PUB KEY
181
CSP VERIFIER
174
CSP
179
BINDER'S SIGNATURE
182
KEY MANAGER
184
SIGNING PUB/PRI KEYS
186
KEY EXCHANGE PUB/PRI KEYS
188
SYMMETRIC KEY GENERATOR
190
SYMMETRIC KEY ENCRYPT/DECRYPT
192
ASYMMETRIC KEY ENCRYPT/DECRYPT
194
HASHING CALCULATOR
196
SIGNING DEVICE
176
PAPI

Fig 10

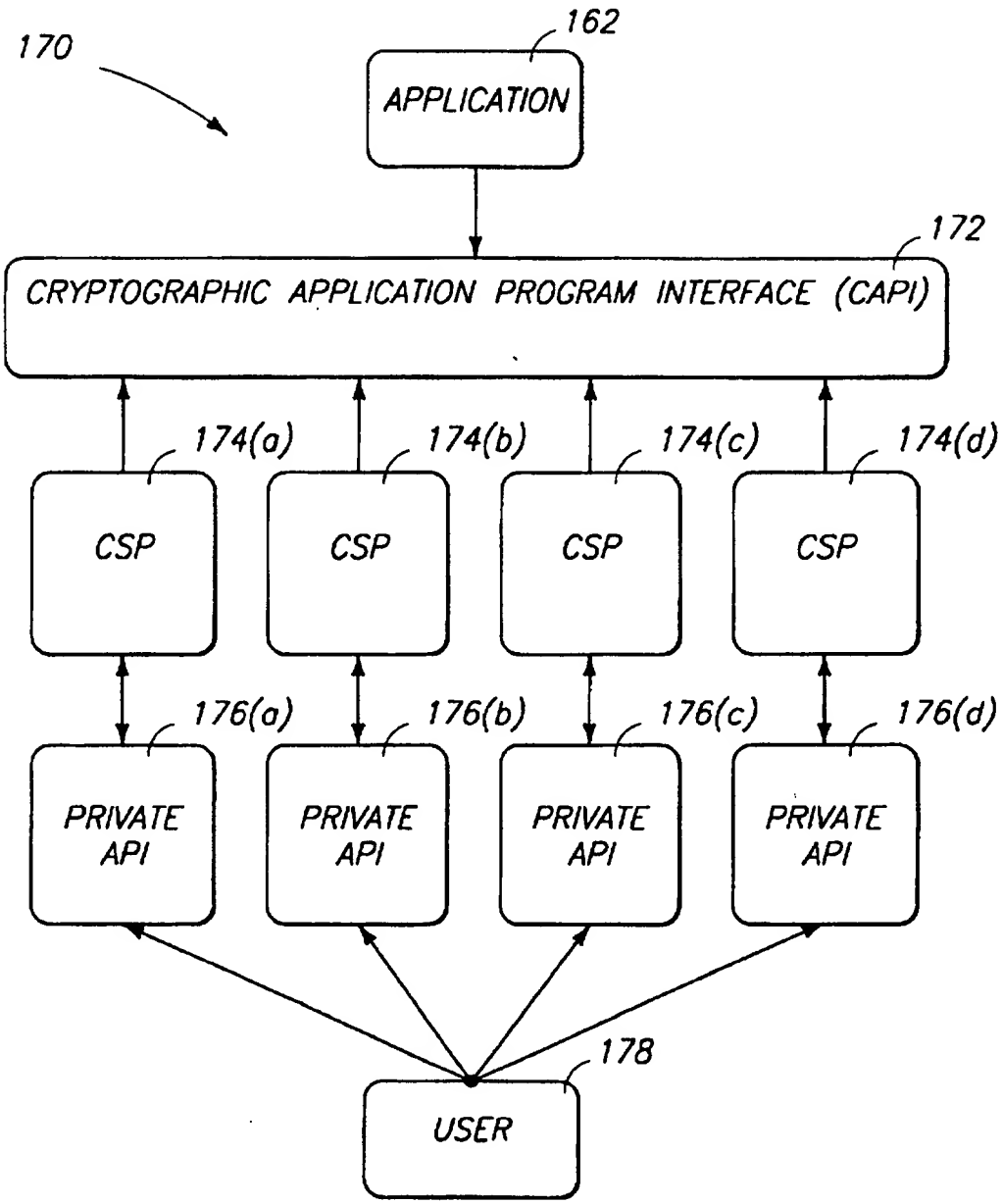

170
162
APPLICATION
172
CRYPTOGRAPHIC APPLICATION PROGRAM INTERFACE (CAPI)
174(a)
174(b)
174(c)
174(d)
CSP
CSP
CSP
CSP
176(a)
176(b)
176(c)
176(d)
PRIVATE API
PRIVATE API
PRIVATE API
PRIVATE API
178
USER

Fig 11

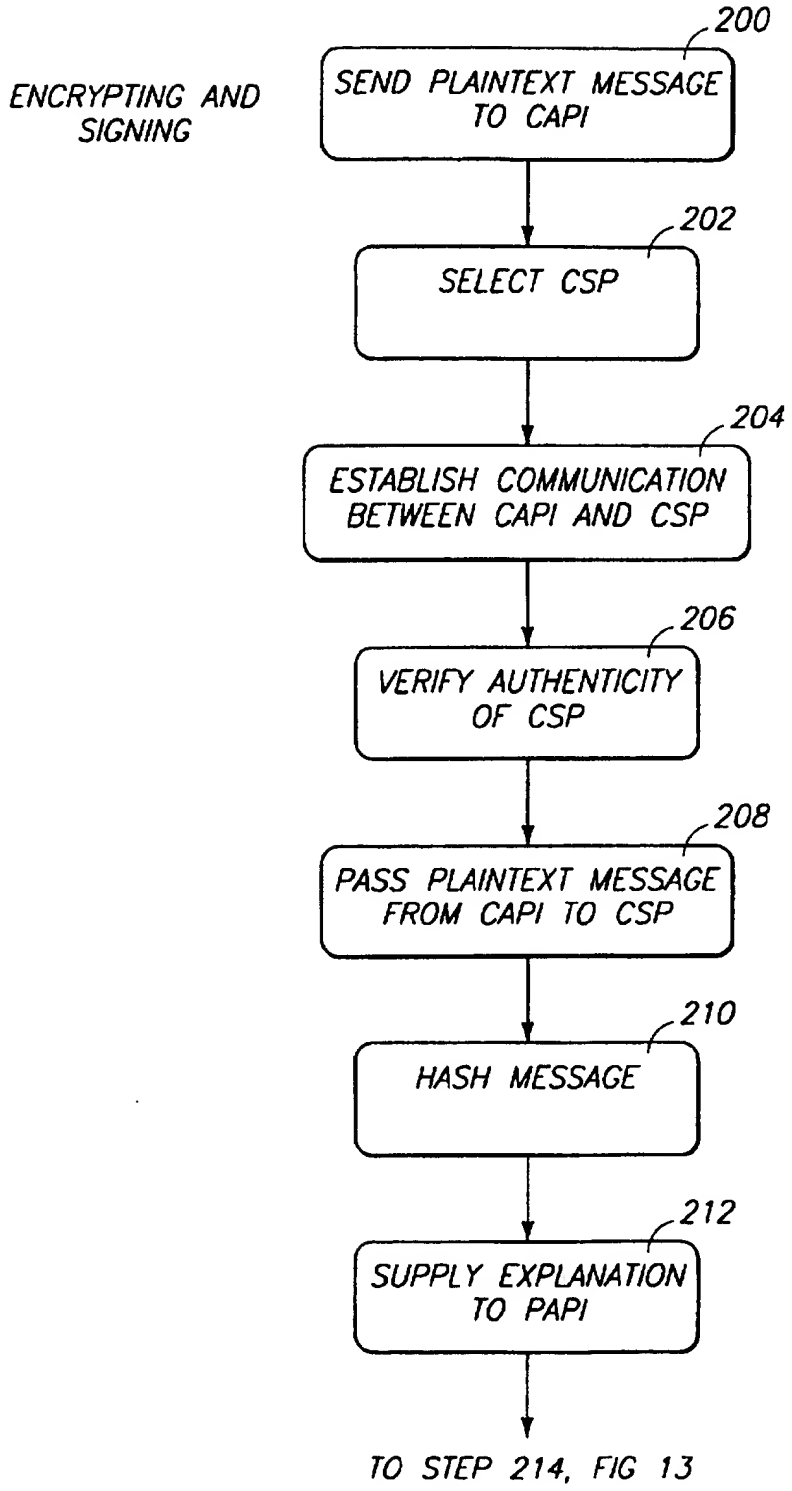

ENCRYPTING AND SIGNING
200
SEND PLAINTEXT MESSAGE TO CAPI
202
SELECT CSP
204
ESTABLISH COMMUNICATION BETWEEN CAPI AND CSP
206
VERIFY AUTHENTICITY OF CSP
208
PASS PLAINTEXT MESSAGE FROM CAPI TO CSP
210
HASH MESSAGE
212
SUPPLY EXPLANATION TO PAPI
TO STEP 214, FIG 13

Fig 12

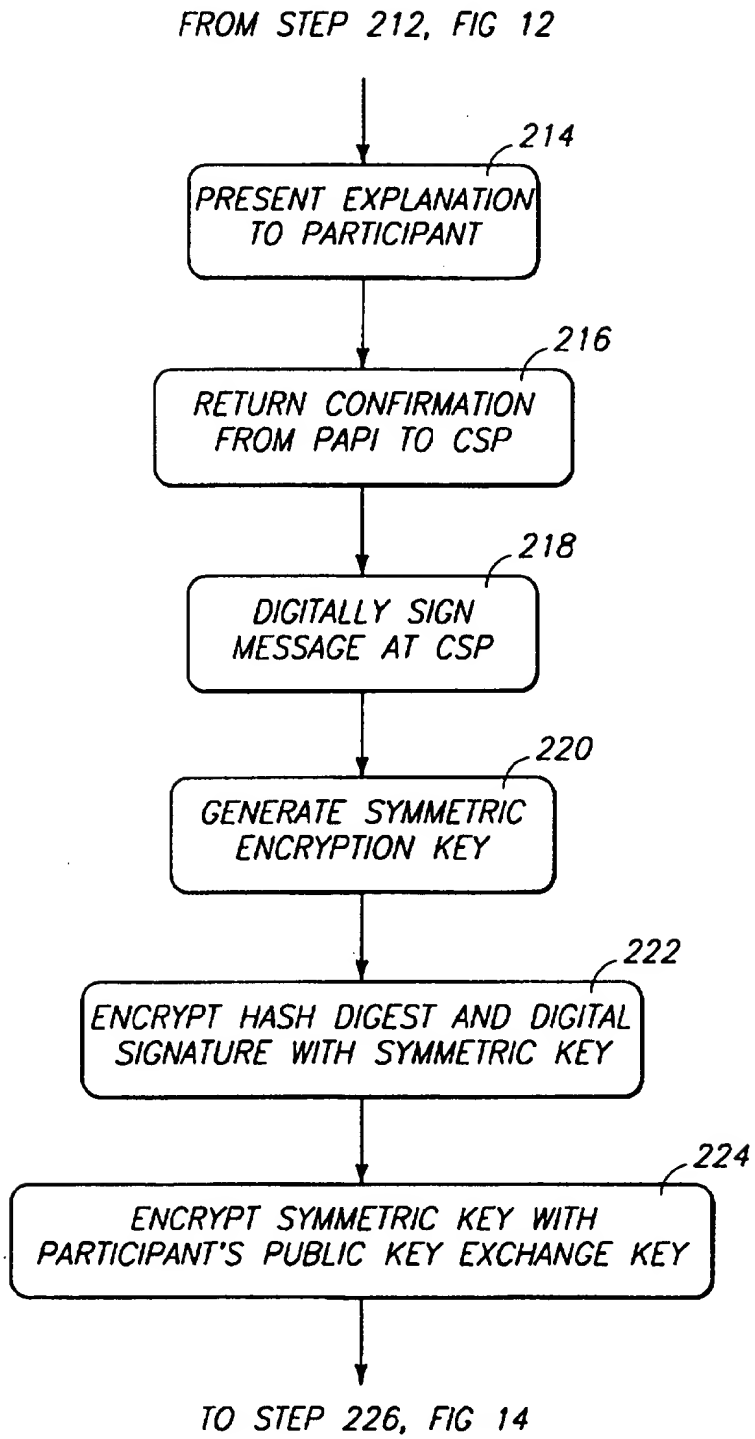

FROM STEP 212, FIG 12
214
PRESENT EXPLANATION TO PARTICIPANT
216
RETURN CONFIRMATION FROM PAPI TO CSP
218
DIGITALLY SIGN MESSAGE AT CSP
220
GENERATE SYMMETRIC ENCRYPTION KEY
222
ENCRYPT HASH DIGEST AND DIGITAL SIGNATURE WITH SYMMETRIC KEY
224
ENCRYPT SYMMETRIC KEY WITH PARTICIPANT'S PUBLIC KEY EXCHANGE KEY
TO STEP 226, FIG 14

Fig 13

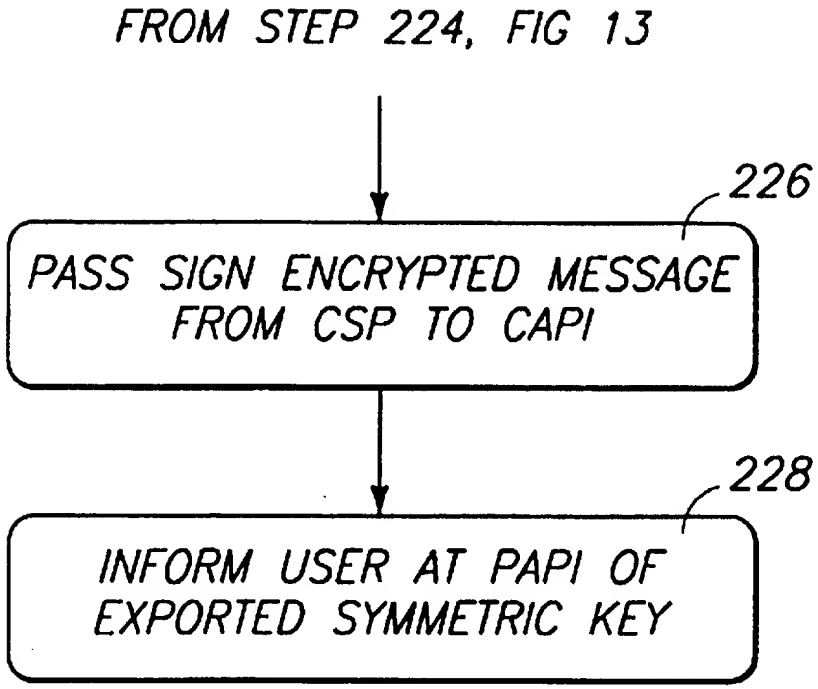

FROM STEP 224, FIG 13
226
PASS SIGN ENCRYPTED MESSAGE FROM CSP TO CAPI
228
INFORM USER AT PAPI OF EXPORTED SYMMETRIC KEY

Fig 14

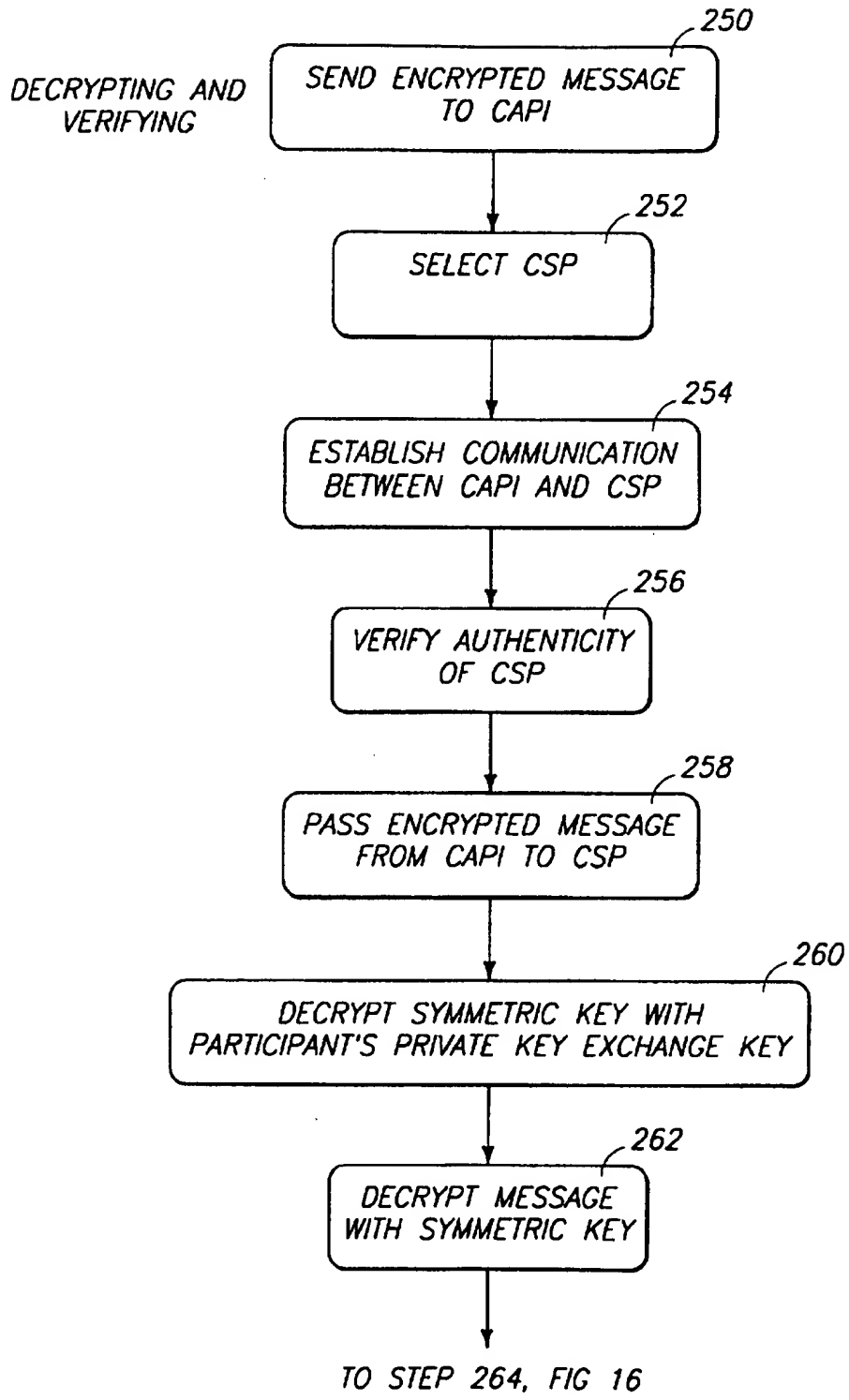

DECRYPTING AND VERIFYING
250
SEND ENCRYPTED MESSAGE TO CAPI
252
SELECT CSP
254
ESTABLISH COMMUNICATION BETWEEN CAPI AND CSP
256
VERIFY AUTHENTICITY OF CSP
258
PASS ENCRYPTED MESSAGE FROM CAPI TO CSP
260
DECRYPT SYMMETRIC KEY WITH PARTICIPANT'S PRIVATE KEY EXCHANGE KEY
262
DECRYPT MESSAGE WITH SYMMETRIC KEY
TO STEP 264, FIG 16

Fig 15

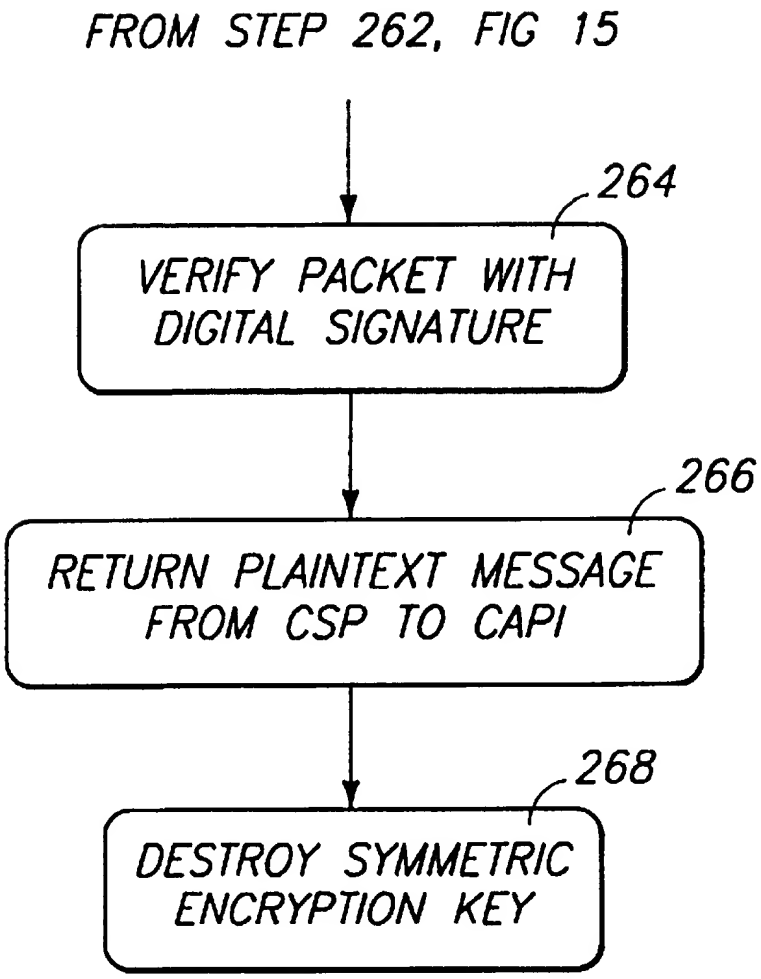

FROM STEP 262, FIG 15
264
VERIFY PACKET WITH DIGITAL SIGNATURE
266
RETURN PLAINTEXT MESSAGE FROM CSP TO CAPI
268
DESTROY SYMMETRIC ENCRYPTION KEY

Fig 16

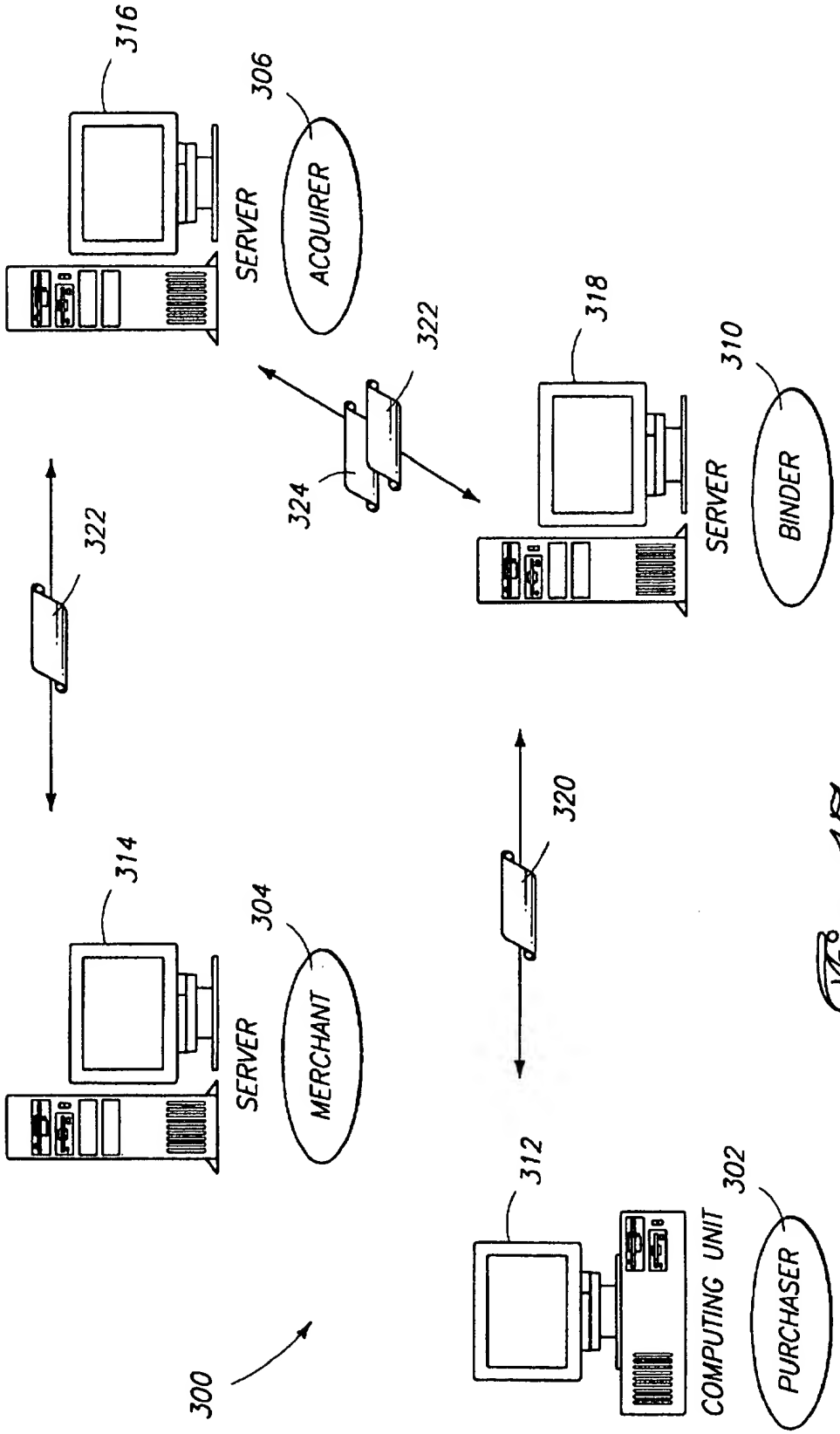

316
306
SERVER
ACQUIRER
322
318
324
322
310
SERVER
BINDER
322
320
314
304
SERVER
MERCHANT
312
302
COMPUTING UNIT
PURCHASER
300

Fig 17

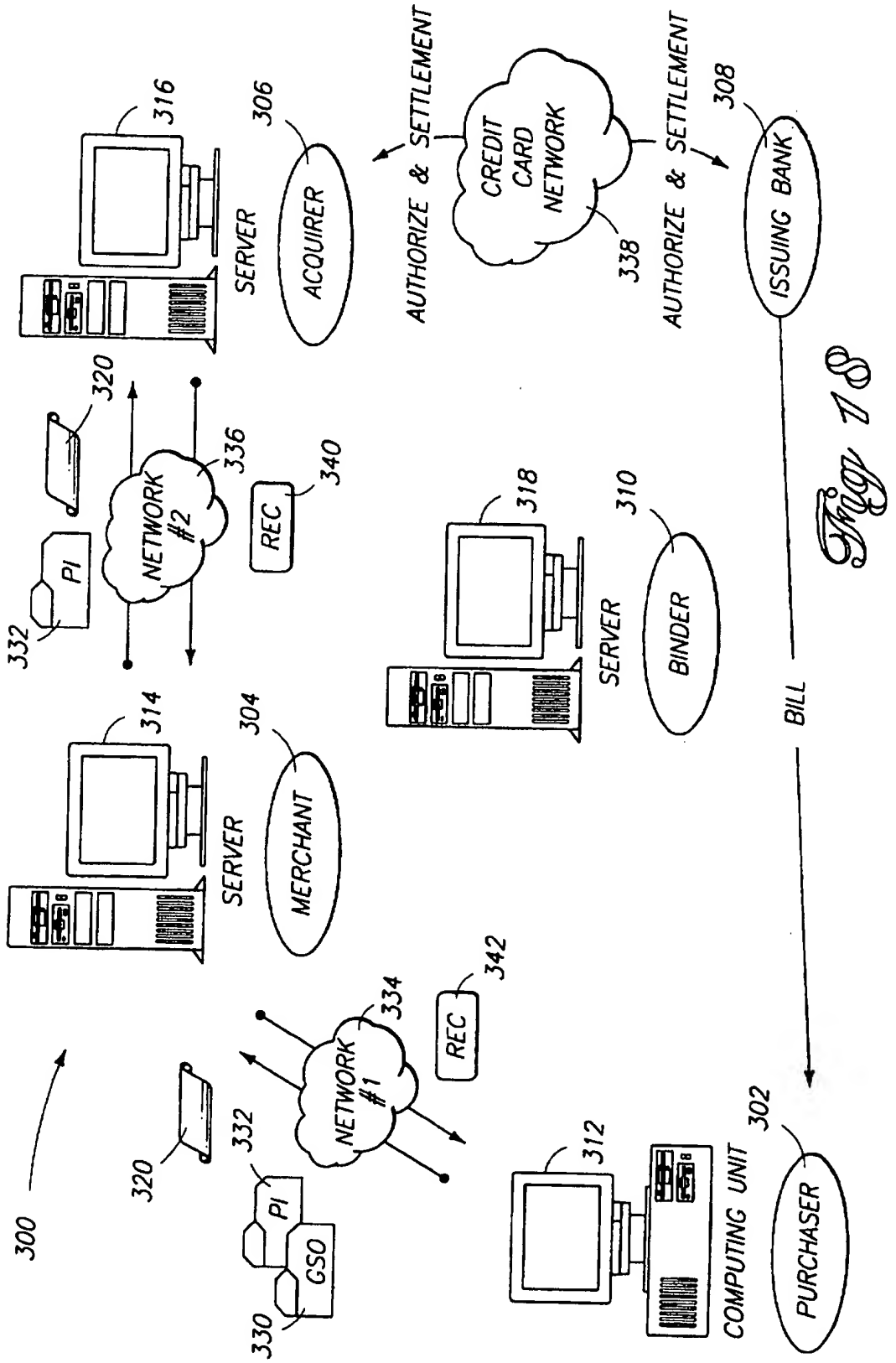

300
316
306
SERVER
ACQUIRER
AUTHORIZE & SETTLEMENT
CREDIT CARD NETWORK
338
AUTHORIZE & SETTLEMENT
308
ISSUING BANK
320
336
340
NETWORK #2
REC
PI
332
318
310
SERVER
BINDER
314
304
SERVER
MERCHANT
BILL
342
334
REC
NETWORK #1
320
332
PI
GSO
330
312
302
COMPUTING UNIT
PURCHASER

Fig 18

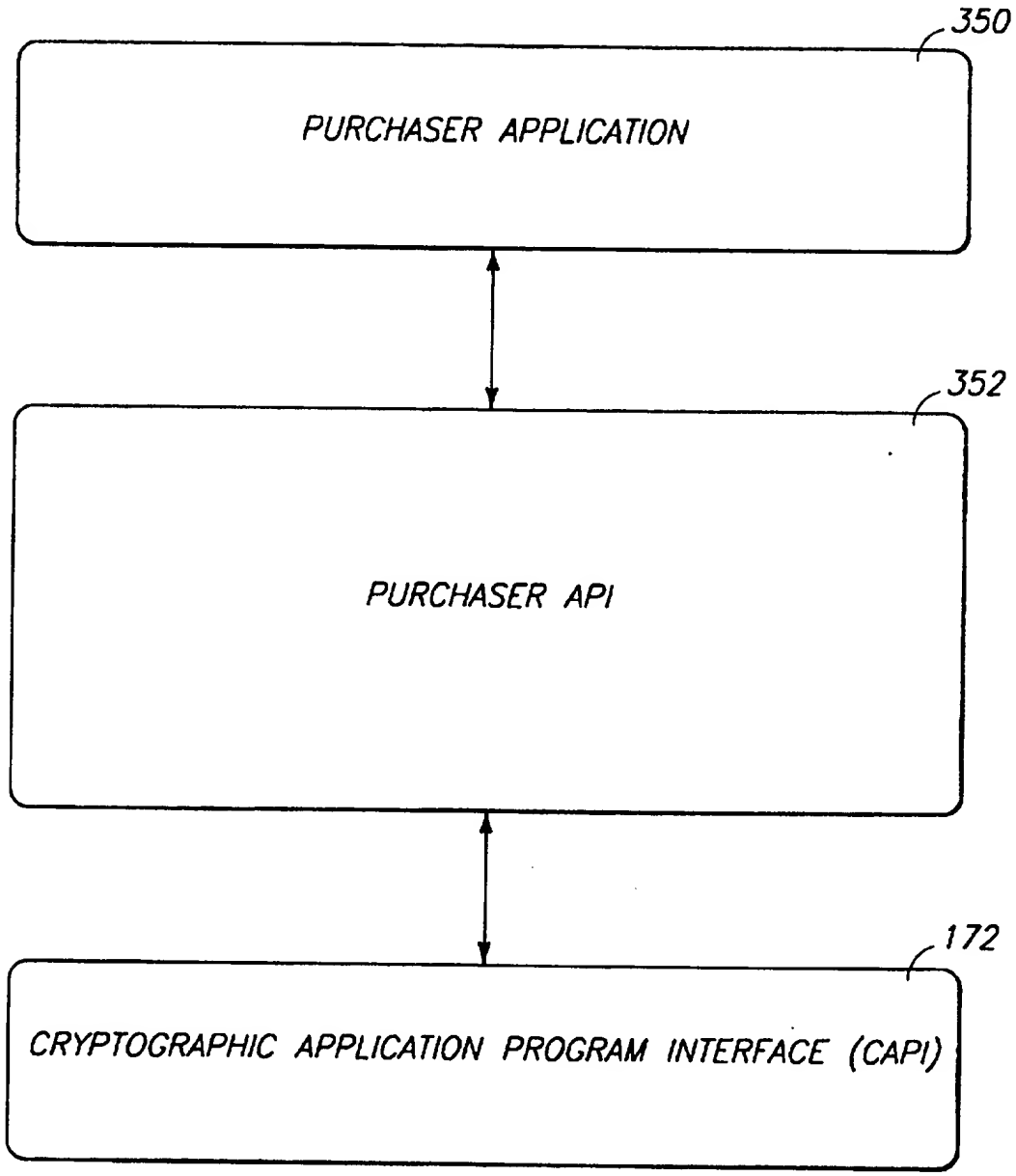

350
PURCHASER APPLICATION
352
PURCHASER API
172
CRYPTOGRAPHIC APPLICATION PROGRAM INTERFACE (CAPI)

Fig 19

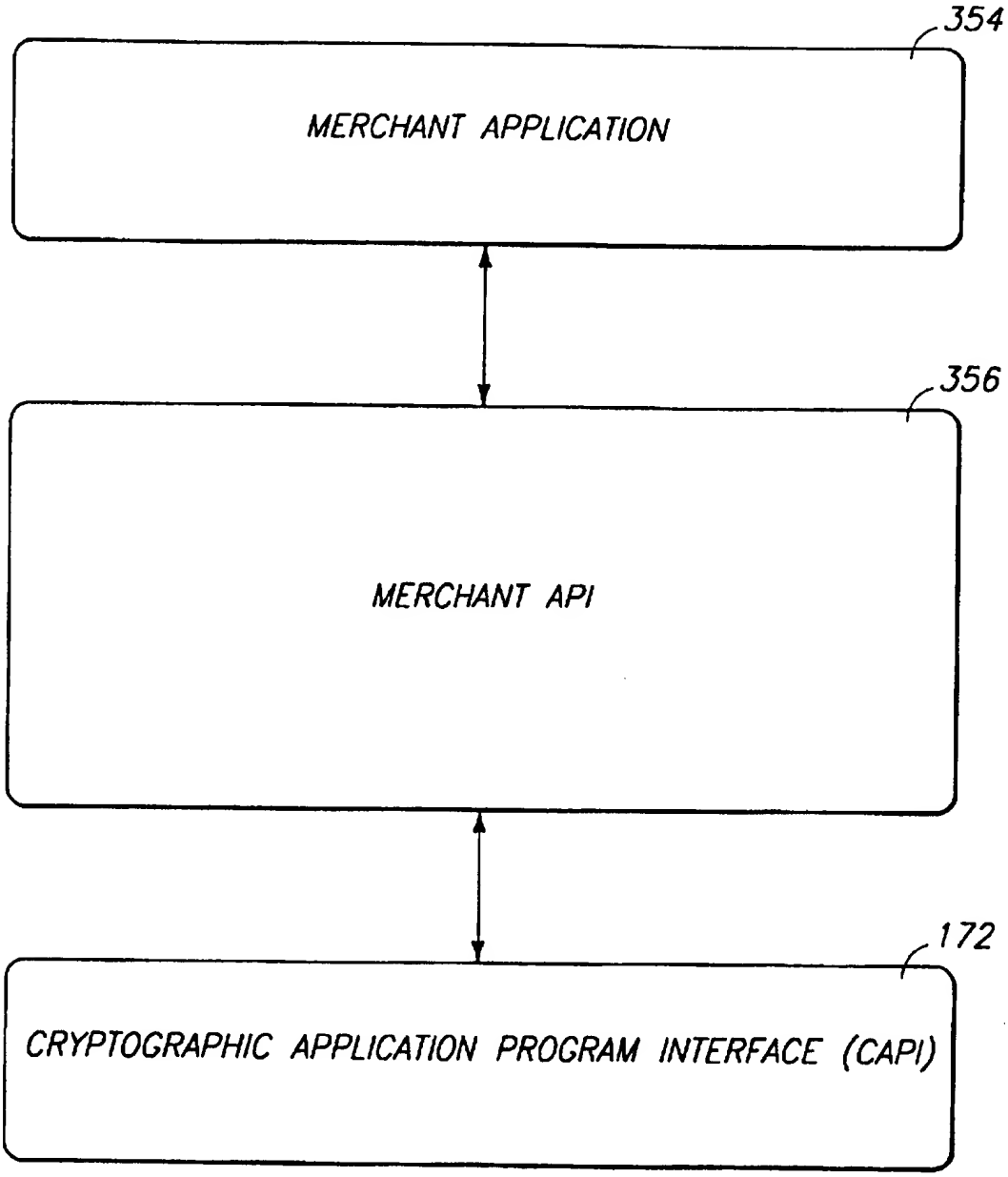

354
MERCHANT APPLICATION
356
MERCHANT API
172
CRYPTOGRAPHIC APPLICATION PROGRAM INTERFACE (CAPI)

Fig 20

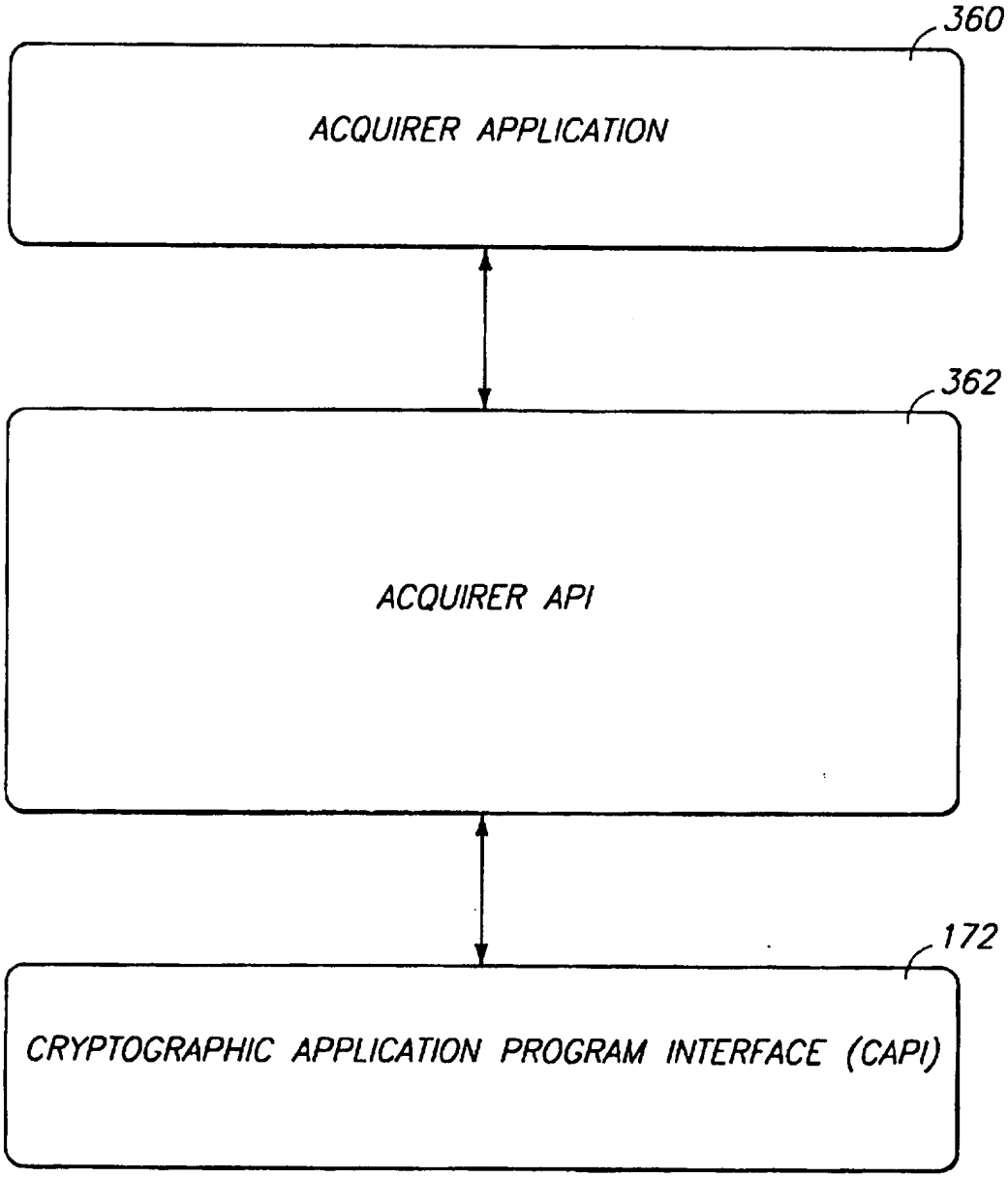

360
ACQUIRER APPLICATION
362
ACQUIRER API
172
CRYPTOGRAPHIC APPLICATION PROGRAM INTERFACE (CAPI)

Fig 21

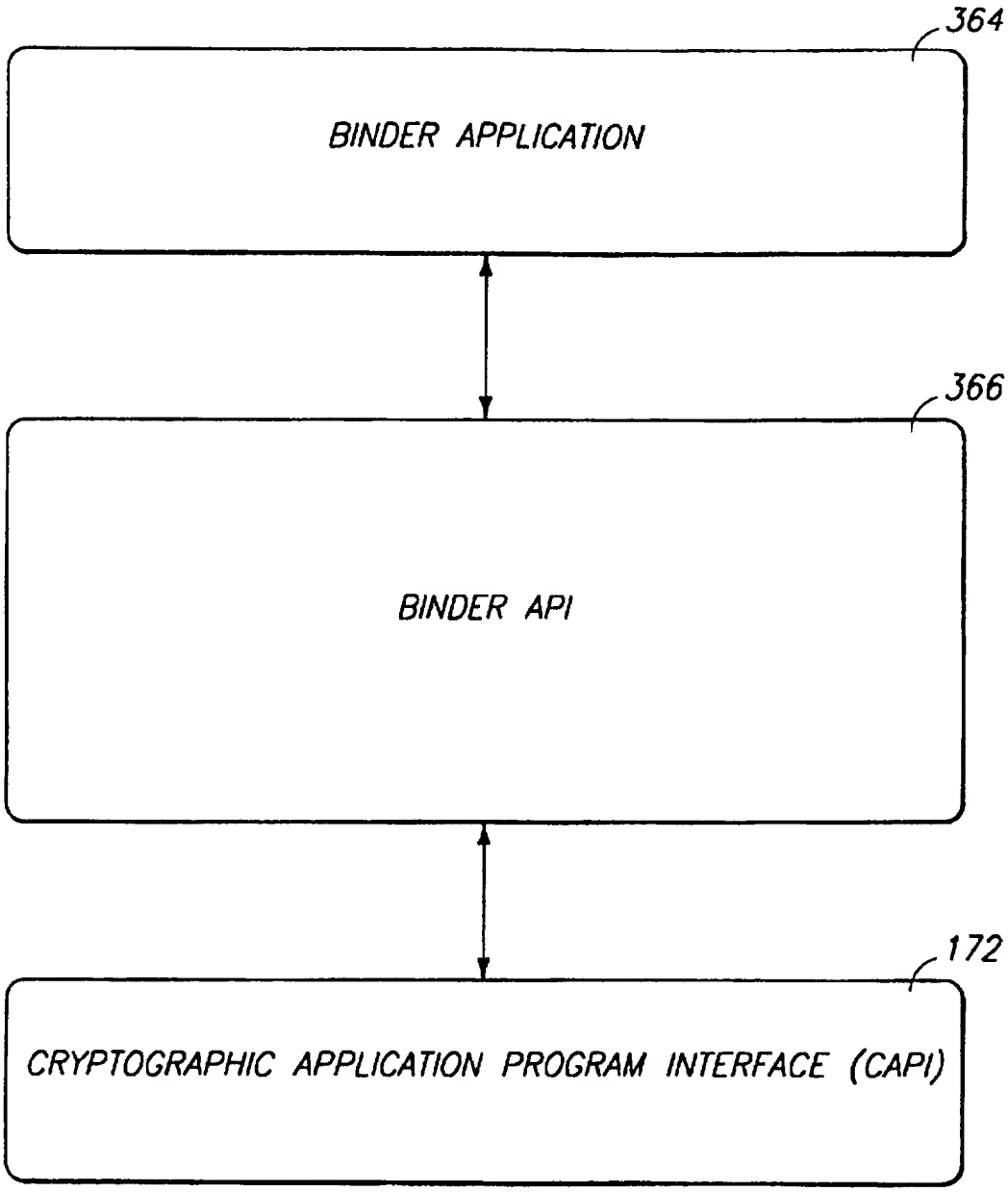

364
BINDER APPLICATION
366
BINDER API
172
CRYPTOGRAPHIC APPLICATION PROGRAM INTERFACE (CAPI)

Fig 22

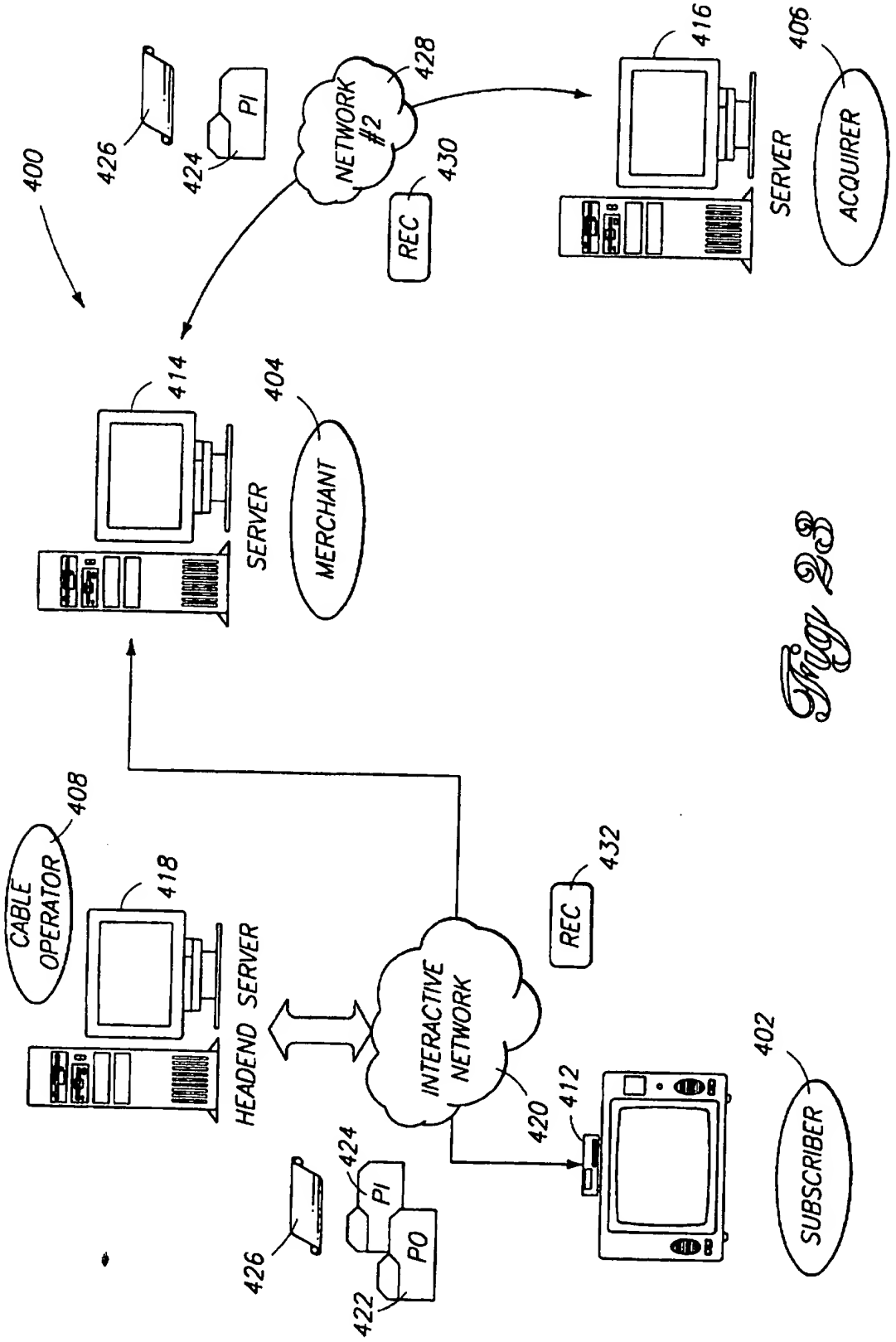

400
426
424
PI
NETWORK #2
428
430
REC
416
406
SERVER
ACQUIRER
414
404
SERVER
MERCHANT
408
CABLE OPERATOR
418
HEADEND SERVER
INTERACTIVE NETWORK
432
REC
420
412
402
SUBSCRIBER
424
PI
PO
426
422

Fig 23

CRYPTOGRAPHY SYSTEM AND METHOD FOR PROVIDING CRYPTOGRAPHIC SERVICES FOR A COMPUTER APPLICATION

TECHNICAL FIELD

This invention relates to cryptography systems. More particularly, this invention relates to a computer implemented architecture for performing cryptographic primitives including encrypting, decrypting, signing and verifying/authenticating functions.

BACKGROUND OF THE INVENTION

Cryptography is the an and science of keeping messages secure from eavesdroppers and adversaries. Historically, valuable messages were kept secure by personal envoys who hand carried sensitive information from a sending party to a receiving party. While useful in its time, this protection method is not very practical in a modem world where information flows freely and changes rapidly.

In more recent history, with the advent of computers, wireless communication, and other technological advances, information can be exchanged very quickly among many different individuals who were often spread all over the world. To provide a secure interchange of information in the electronic arena, one traditional approach to mitigating the risk of having sensitive information intercepted was to institute proprietary computerized systems that were closed to the general public. Such proprietary systems promoted security simply by restricting physical access through high security protocols. A private communication network linked only those terminals that were authorized, and only participants to the system with the appropriate security clearance were permitted access to the terminals. Hence, participants and information were authenticated by definition, and the integrity and value of the information were preserved within the confines of the closed processing system. Unfortunately, proprietary systems are not useful in a grander context which envisions the interchange of information among virtually any individuals without limitation.

Cryptography has evolved in the electronic setting as a means to securely transfer information over a communication system that is presumed to be insecure, like the telephone lines or a public communications network (e.g., the Internet). In this electronic computerized context, cryptography provides the necessary tools to digitally secure sensitive and valuable electronic messages in a manner that insures privacy between the authenticate sender and authenticate recipient of the communiqué, even though the message is subject to interception on the insecure communication system.

Before sending an electronic message, the sender encrypts it. "Encryption" transforms the message from its plaintext into some meaningless ciphertext that is not understandable in its raw form and cannot be deciphered by an eavesdropper. To ensure the recipient that the true sender originated the message, and not some impostor, the sender "digitally signs" the message with its own unique digital signature. The signed encrypted message is then transmitted over the insecure network to the intended recipient. The recipient receives and decrypts the encrypted message. "Decryption" transforms the message from its ciphertext back to its plaintext. Only the recipient is presumed to have the ability to decipher the message. The recipient also "verifies" the authenticity of the digital signature to assure itself that the contents are from the legitimate sender and have not been subsequently altered.

Encryption, decryption, digital signing, and verification are principal cryptographic primitives that are used in an electronic network setting to facilitate the security, privacy, authenticity, and integrity of information being exchanged. These cryptographic primitives commonly involve the use of secret cryptographic keys. "Keys" are a numerical value, often expressed digitally as a number of bits, which are used in the cryptographic algorithms that encrypt and decrypt messages. The keys are uniquely associated with a particular identity, such as a person, group, physical object, business, or institution. The keys are kept secret and used selectively by the identity to perform the cryptographic primitives as required. For example, a person might use a key to encrypt and sign a purchase order message intended for a merchant. The merchant might then use a key to decrypt the message and verify the authenticity of the signature.

In a network setting, it is desirable to locate cryptographic functions in the computer operating system so that they are available to other applications executing on the computer. Furthermore, it is desirable to define an application program interface (API) which allows the applications access to the cryptographic functions in a standardized way.

A cryptographic API raises a number of important security issues. Cryptographic functions, such as encryption and signing, can be performed using a number of different algorithms and formats. Security can vary widely among the different approaches. A single module of the operating system cannot possibly implement all possible algorithms and formats that a user or application might want to use in a given situation.

On the other hand, cryptographic functionality involves the handling of sensitive information and the maintenance of secure keys. It would be advantageous for a user or application to be able to trust the cryptographic system on the same level that an operating system is typically trusted.

There are also potential hazards of using cryptographic functions in the computerized network setting. Since the functions are carried out electronically, the user might assume the cryptographic routines are operating as expected, yet not be aware of ignorant or sophisticated electronic attacks. Careless applications might use cryptographic encryption or signature keys in ways that jeopardize the keys' secrecy. Moreover, malicious applications might even deliberately compromise the user's secrecy, or worse, perform unauthorized cryptographic operations. For instance, a malicious application might attempt to decrypt the user's secret files and transmit them to some adverse party. Another situation might involve an application attempting to digitally sign notes or IOUs on behalf of the user without the user's knowledge or consent. A computer implemented cryptographic system must therefore provide the needed security to prevent attack from poorly devised or malicious applications.

Today, there are several electronic systems that provide cryptographic services in the computer forum. These include "Bsafe libraries" by RSA Data Security Inc., "X/Open CAPI", and "PKCS#". However, each of these systems permit direct access of the application to keying material. There is no protection of these cryptographic resources from electronic attack. Furthermore, the Bsafe system, which is the most widely used cryptography system, directly attaches the cryptographic code to the application. There is no contemplation of protecting the cryptographic functions within the computer from ignorant or malicious attacks from other software applications.

It would therefore be advantageous to provide a computer implemented architecture for performing cryptographic

primitives that maintains and protects the user's keys and prevents undesired access and use of cryptographic functions without authorization from the user.

SUMMARY OF THE INVENTION

This invention provides a cryptographic system and method that protect a user's keys and prevents undesired access and use of cryptographic functions without authorization from the user. The cryptographic system is a unique tri-layer architecture. It includes a cryptographic application program interface (CAPI) which provides functionality to an application, one or more cryptographic service providers (CSPs) which implement the functionality presented by the CAPI to the application, and one or more private application program interfaces (PAPI) which allow the CSPs to communicate directly with a user.

The CAPI layer provides the interface with an application that requests cryptographic functions such as encryption, decryption, signing, or verification. The CAPI selects the appropriate CSP for performing the requested cryptographic function. The CSPs perform the cryptography functions and manage the cryptographic keys used in the functions. For instance, one or more CSPs might be configured to perform the encryption function using certain types of cryptographic algorithms and keys; one or more different CSPs might be configured to decrypt the messages; another CSP might be embodied to digitally sign messages; while still another CSP might be designed to verify signatures and messages. Preferably, the CSPs are implemented as dynamic linked libraries (DLLs) that are loaded on demand by the CAPI, and which can then be called by the application through the CAPI.

The CAPI also authenticates the chosen CSP. Each CSP has a digital signature of a trusted authority which can be authenticated by the CAPI to ensure that impostor CSPs are not introduced to the system.

To promote security, the application is restricted from direct access to the cryptographic keys maintained in the CSPs, but is permitted to manipulate the keys through the use of handles assigned by the CSPs. Cryptographic keys used in the encryption and decryption are not exported from the CSPs in their raw form, but only in an encrypted form. Moreover, certain confidential private keys are not permitted to leave the CSPs under any circumstances. In this manner, the CSPs protect the keys from compromise, while simultaneously offering a full range of cryptographic functionality that the user/application requests.

The PAPI keeps the user informed as to the functions being performed by the CSPs. The PAPI provides the user with direct access to the CSP, independent of the application. The PAPI can present information to the user that the user may not trust the application to provide. For instance, the CSP can present through the PAPI an explanation of the transaction being conducted by the CSP. The CAPI might also present a dialog box asking for the user's confirmation each time an application requests a digital signature to enable the user to confirm or reject digitally signing the message. The PAPI might also be configured to notify the user and seek permission to export certain cryptographic keys.

This system architecture can be advantageously adapted to many different environments. By implementing the cryptographic services as independent and separate CSPs that are only accessible through a CAPI layer, the architecture affords maximum protection of sensitive cryptographic keys. Additionally, by implementing the CSPs as DLLs, the cryptographic functions themselves and the associated security levels can be easily modified or replaced without affecting the higher level application. This is useful for rapid conformation to acceptable and changing regulatory and legal practices imposed by various governments. Finally, by providing the PAPI layer, the architecture protects the user from malicious applications that attempt to expose key information or gain unauthorized signatures of the user.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** is a schematic of an electronic commerce system during a registration process according to one aspect of this invention.

FIG. **2** is a schematic of the electronic commerce system during a transaction process according to another aspect of this invention.

FIG. **3** is a flow diagram of the registration process in a method for conducting an electronic commerce transaction according to yet another aspect of this invention.

FIG. **4** is a flow diagram of a process for generating a registration packet that is performed during the registration process.

FIG. **5** is a flow diagram of a process for verifying a registration packet that is performed during the registration process.

FIGS. **6–8** present a flow diagram of the transaction process in the method for conducting an electronic commerce transaction.

FIG. **9** is a data structure used in the interchange of data between participants in the electronic commerce system.

FIG. **10** is a block diagram of a computing unit provided at each participant in the electronic commerce system.

FIG. **11** is a block diagram of an architecture for a cryptography system according to yet another aspect of this invention.

FIG. **12–14** are a flow diagram of encrypting and signing functions performed by the cryptography system.

FIG. **15–16** are a flow diagram of decrypting and verifying functions performed by the cryptography system.

FIG. **17** is a schematic of the electronic commerce system embodied as a credit card system according to another aspect of this invention. FIG. **17** shows the credit card system is during the registration phase.

FIG. **18** is a schematic of the credit card system during the transaction phase.

FIGS. **19–22** are block diagrams depicting the FIG. **11** cryptography system as services to operating system applications resident at a purchaser, merchant, acquirer, and binder participants to the FIG. **18** credit card system.

FIG. **23** is a schematic of the electronic commerce system embodied as an interactive entertainment system according to yet another aspect of this invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

The following discussion assumes that the reader is familiar with cryptography. For a basic introduction of cryptography, the reader is directed to a text written by Bruce Schneier and entitled "Applied Cryptography: Protocols, Algorithms, and Source Code in C," published by John Wiley & Sons with copyright 1994, which is hereby incorporated by reference.

This invention particularly concerns a cryptographic system architecture and method for providing cryptographic

functionality in a computer network environment. Primary aspects of this invention are described in detail with reference to FIGS. **10–16**. The cryptographic system can be implemented in any context that requires cryptographic services, including such functions as encryption, decryption, digital signing, and authentication.

To provide an example context, however, aspects of this invention are described in the context of an electronic commerce system which facilitates the secure interchange of commercial documents and instruments over an insecure communication system. By describing the invention in a particular example context, the reader will appreciate how the unique architecture and methodology can be incorporated into a practical setting. From this, the reader will better understand the invention and how it can be implemented in essentially any electronic environment which has need for cryptographic functions. The following sections describe the example context of an electronic commerce system.

Electronic Commerce System

FIGS. **1** and **2** show an electronic commerce system **20** for conducting secure electronic commerce transactions. The electronic commerce system **20** includes multiple trading partners or participants, which are represented by three participants **22**(*a*), **22**(*b*), and **22**(*c*), and a certified trusted authority **26**. Each individual electronic commerce transaction involves at least one commerce instrument and at least one commerce instrument. The commerce document defines the type of commerce transaction. Examples of commerce documents include purchase orders and receipts, contracts, and payment instruction receipts. The commerce instrument defines a mode of payment for the transaction. Examples of commerce instruments include payment instructions (e.g., checks and credit cards) and currency.

Computing units **24**(*a*), **24**(*b*), and **24**(*c*) are provided at respective ones of the participants **22**(*a*), **22**(*b*), and **22**(*c*). The computing units are depicted for illustration purposes as IBM®-compatible personal computers, although other forms of computing units may be used. For instance, the computing units might be embodied as conventional computers (such as mainframe computers, servers, PCs, laptops, notebooks, etc.) or as other computational machines, such as banking ATMs (automated teller machines) and set-top boxes used in an interactive television system.

A computer server **28** is provided at the certified trusted authority **26**. In the commercial context, the certified trusted authority is often referred to as a "credential binder," "binding authority," or simply "binder." The server **28** is thus referred to herein as the "credential binding server." The computer server is capable of receiving simultaneous requests from the multiple participants, as will be described in more detail below.

The computing units **24**(*a*), **24**(*b*), and **24**(*c*) and computer server **28** are interconnected with each other via one or more communication systems. The communication systems can be embodied as a wire-based or wireless network. Examples of communications systems include an ATM (asynchronous transfer mode) switching network, a public network, a wide area network, an interactive television (ITV) network, a credit card network, a satellite network, and an RF network.

The electronic commerce system of this invention is extremely flexible and can be adapted to many different commerce transactions, as will become more evident from the continuing discussion. In the general model of the electronic commerce system **20** shown in FIGS. **1** and **2**, a "participant" can be an individual person (such as a credit card holder, an ITV viewer, or a banking member at an automated teller machine), an entity or business (such as a merchant, cable operator, or service provider), or an institution (such as a bank). The certified trusted authority is a third party entity that every participant thoroughly trusts. This authority is established and recognized by all involved. Example certifying authorities in the financial environment include the federal reserve or a bank. The commerce environment in which the electronic commerce system is implemented defines the rules and information required to carry out the transactions, and additionally defines who or what entity is the certified trusted authority.

The electronic commerce system **20** can be implemented in different environments. As one example, the electronic commerce system can be implemented in an credit card network system. The electronic commerce system employs essentially the same existing banking and credit card commerce structure, while broadening access to that system to individual consumers. This example implementation is described below in more detail with reference to FIGS. **17** and **18**. As another example, the electronic commerce system can be implemented in an interactive entertainment system which supports set-top boxes in each subscriber's home. This example implementation is described below in more detail with reference to FIG. **23**.

The electronic commerce system **20** facilitates the secure exchange of commerce documents and commerce instruments over a communications system that is assumed to be insecure and open to eavesdroppers. Each computing unit is capable of encrypting or decrypting messages, digitally signing the messages, and verifying the authenticity of the messages from other participants. One preferred implementation of a cryptography system architecture is described below with reference to FIGS. **10** and **11**.

General Operation

There are two distinct phases or processes that are performed by the electronic commerce system to provide the secure document interchange. The first phase, illustrated in FIG. **1**, is an initial registration process in which each participant seeks approval of the certified trusted authority. The second phase, illustrated in FIG. **2**, is a transaction process which involves the general document and instrument interchange among the various participants.

During the registration process (FIG. **1**), the computing units **24**(*a*)–**24**(*c*) at the participants **22**(*a*)–**22**(*c*) are each programmed to generate and send a registration packet over the communication system (as represented by communication paths **30**(*a*)–**30**(*c*)) to the credential binding server **28** at the trusted credential authority **26**. The credential binding server **28** is programmed to produce unique credentials for each participant based upon their registration packets and to send the credentials **32**(*a*)–**32**(*c*) back over the communication system (as represented by communication paths **34**(*a*)–**34**(*c*)) to the multiple computing units **24**(*a*)–**24**(*c*). These credentials are digitally signed by the trusted credential authority and will be used to identify and authenticate other participants during the commerce transaction. It is noted that the registration process requires interaction between each participant and the trusted credential authority.

After registration, the participants are ready to conduct their commercial activity. As shown in FIG. **2**, the transaction process involves communication among the participants to the transaction without any interaction between the participants and the trusted credential authority. This is beneficial because it eliminates the need to check with a trusted credential authority during each commercial transaction and streamlines communication during the transaction.

Each commerce transaction has at least one originating participant and one or more recipient participants. A computing unit **24**(*a*) at the originating participant **22**(*a*) is programmed to request and receive the credentials of all intended recipient computing units **24**(*b*) and **24**(*c*). The originating computing unit also verifies the credentials by checking the digital signature of the trusted credential authority. The originating computing unit **24**(*a*) then generates commerce document(s) **36** and commerce instrument(s) **38** that are appropriate for the type of commercial transaction. The document(s) and instrument(s) are both encrypted and sent together over a communication path **40** to the computing unit **24**(*b*) at the first recipient participant **22**(*b*). The document(s) and instrument(s) are encrypted using appropriately different keys so that only the participants to whom the document(s) or instrument(s) pertains can decrypt them.

The first recipient computing unit **24**(*b*) is programmed to decrypt either the commerce document(s) **36** or the commerce instrument(s) **38** depending upon which one(s) is intended for and pertains to them. The first recipient computing unit **24**(*b*) then passes the other(s) of the commerce document(s) **36** or the commerce instrument(s) **38** in encrypted form over a communication path **42** to a second computing unit **24**(*c*) at the second recipient participant **22**(*c*). The second recipient computing unit is programmed to decrypt the document(s) or instrument(s) intended for and pertaining to them.

As an example, the originating participant is a consumer, the first recipient participant is a merchant, and the second recipient participant is a financial institution, like a bank. The commerce document **36** is a purchase order from the consumer to buy goods or services from the merchant. The commerce instrument **38** is a payment instruction to the bank regarding how the consumer desires to pay for the goods or services. The merchant's computing unit **24**(*b*) decrypts the commerce document **36** (i.e., a purchaser order) which pertains to the purchase of goods or services from the merchant. The bank's computing unit **24**(*c*) decrypts the commerce instrument (i.e., the payment instructions) which pertains to payment for those goods or services. For added security, key components of the payment instruction (such as account number) may be encrypted within the public key envelope (i.e., next to the symmetric key used to encrypt the rest of the payment instruction).

Assuming the second recipient participant **22**(*c*) can satisfy the commerce instrument, the computing unit **24**(*c*) returns a signed authorization receipt **44** over communication path **46** to the first recipient participant **22**(*b*) indicating that payment is guaranteed. The first recipient computing unit **24**(*b*) then sends a signed purchase receipt **48** over communication path **49** to the originating computing unit **24**(*a*) and complies with the commerce document in delivering the ordered goods and services.

The electronic commerce system of this invention is advantageous because it optimizes efficiency in communication between the participants. During the transaction, the originating participant sends one package containing both the document and instrument to the same recipient. The document and instrument are encrypted differently so that only the intended recipient can open them. This aspect promotes security. The first recipient decrypts that portion pertaining to them, and forwards the rest on to the next recipient. Again, the first recipient only sends a package to one party. It is further noted that each participant can verify the contents using the digital signatures of the originating participant, and the digital signature of the trusted authority. In this manner, each participant can be assured that it is dealing with an authenticated party, without having to check with the trusted authority each time. Further, the first party can confirm that the goods/services can be sent/rendered before involving the bank.

The electronic commerce system and method for operating it will now be described in more detail with reference to FIGS. **1–8**. The registration process is described with reference to FIG. **1** and the flow diagram of FIGS. **3–5**, and the transaction process is described with reference to FIG. **2** and the flow diagram of FIGS. **6–8**.

Registration Process

Each participant **22**(*a*), **22**(*b*), and **22**(*c*) is required to register with the certified trusted authority **26** before being permitted to engage electronically in the commercial activity. The registration process involves communication over the communications system between each participant and the certified trusted authority as illustrated in FIG. **1**. Each participant generates a registration packet containing general information about that participant (step **50** in FIG. **3**). An application tailored to the particular commerce environment can be distributed to the participants to assist them in gathering and submitting the information required by the certified trusted authority. For example, the registration packet includes identification information (name, location, etc.), public cryptography keys unique to the participant, and a digital signature of the participant.

FIG. **4** shows the method for generating the registration packet according to a preferred implementation. At steps **70** and **72**, the computing unit generates two asymmetric pairs of public and private cryptography keys. An "asymmetric" key algorithm involves two separate keys, for example, one key to encrypt and one key to decrypt. The keys are based upon a mathematical relationship in which one key cannot be calculated (at least in any reasonable amount of time) from the other key. Encryption and decryption using an asymmetric key pair can be represented as follows:

$$E_{Kpri}(M)=M'$$

$$D^{Kpub}(M')=M$$

where "E_{Kpri}" is an encryption function using a private key "Kpri," "M" is a plaintext message, "M'" is an encrypted version of the plaintext message, and "D^{Kpub}" is a decryption function using the public key "Kpub". The inverse is also true in that a plaintext message can be encrypted using the public key and then decrypted using the private key.

In a public key system, the public key is distributed to other parties and the private key is maintained in confidence. The asymmetric public and private keys ensure two results. First, only the holder of the private key can decrypt a message that is encrypted with the corresponding public key. Second, if another party decrypts a message using the public key, that party can be assured that the message was encrypted by the private key and thus originated with someone (and presumably the holder) of the private key. An example asymmetric cipher is the well-known RSA cryptographic algorithm named for the creators Rivest, Shamir, and Adleman.

At step **70** of FIG. **4**, the computing unit generates a signing pair of public and private cryptography keys which are used during commerce transactions to generate the participant's digital signature by encrypting the output of a hash function. At step **72**, the computing unit generates an asymmetric key exchange pair of public and private cryptography keys. The key exchange pair is used during the

commerce transaction to encrypt keys that in turn are used to encrypt data contained in the messages.

At step **74**, the participant generates a digital signature that is unique to the participant and to the message. The digital signature is computed by hashing the data contained in the registration packet. A hash function is a mathematical function that converts an input data stream into a fixed-size, often smaller, output data stream that is representative of the input data stream. Once the hash is computed, it is encrypted by the computing unit with the private encryption key of the signing pair (step **76** of FIG. **4**). This is represented as follows:

$$E_{Ksign_pri_part}(dS_{part})=dS'part$$

where the "E" denotes an encryption function on the hash of the message "dS_{part}" and the subscript "Ksign_pri_part" means the participant's private key of the signing pair was employed to perform the encryption. By encrypting the participant's signature with its own private signing key, the eventual recipient will be able to verify the participant's digital signature by decrypting the hash using the participant's public signing key, independently computing the hash of the original message, and comparing the locally computed hash with the decrypted hash. The comparison will succeed only if the participant's private signing key was used to encrypt the hash. Since only the originating participant knows the private signing key, the recipient knows that the originating participant actually created the encrypted hash, essentially "signing" the document.

At step **78**, both public keys of the signing pair and the key exchange pair are encrypted with a symmetric cipher using a randomly selected bulk data symmetric encryption key. In a "symmetric" cipher, the encryption key can be calculated from the decryption key, and vice versa. In many cases, the encryption key and the decryption key are the same. Encryption and decryption using a symmetric key can be represented as follows:

$$E_K(M)=M'$$

$$D_K(M')=M$$

where "E_K" is an encryption function using the symmetric key "K" and "D_K" is the decryption function using the same encryption key "K". The symmetric key must be known to both the sender and receiver, but otherwise kept secret. Once the symmetric key is divulged, any party can encrypt or decrypt messages. Example symmetric ciphers are a DES (Data Encryption Standard) encryption algorithm or an RC4 algorithm.

The data to be included in the registration packet is also encrypted with the same, or additional symmetric encryption keys. Step **78** is represented as follows:

$$E_{Ksym}(\text{data}, K_{sign_pub_part}, K_{exc_pub_part})=\text{Data'\& } K'$$

where "E_{Ksym}" is the encryption function using the symmetric key, "$K_{sign_pub_part}$" is the participant's public key of the signing pair, and "$K_{exc_pub_part}$" is the participant's public key of the key exchange pair.

To prevent divulgation of the symmetric key (or keys), the computing unit at the registering participant encrypts the symmetric key using an asymmetric cipher which employs a public encryption key unique to the certified trusted authority or binder (step **80**). By using the binder's public key, the participant is assured that only the binding authority will be able to decrypt the symmetric key, and hence the rest of the registration packet. The encryption of the symmetric key is represented as follows:

$$E_{Kpub_binder}(K_{sym})=K'_{sym}$$

Notice that the data encryption process involves a dual encryption technique: a first encryption of data using the symmetric key, and then a second encryption of the symmetric key using an asymmetric key pair. This dual encryption affords the desired security through the use of a strong asymmetric key pair, while facilitating encryption of bulk data using the more efficient symmetric key.

At step **82** in FIG. **4**, the computing unit places the encrypted data, digital signature, and public keys in a communications packet. The communications packet is suitable for the appropriate communications system supporting the electronic commerce system **20**. Preferably, the communications packet is in the form of individual digital data structures that contain the appropriate routing information to efficiently locate the intended recipient.

With reference again to FIG. **3**, the next step **52** in the registration process is for each participant computing unit **24**(*a*), **24**(*b*), and **24**(*c*) to send its completed registration packet to the credential binding server **28** over the communications system. This step is represented graphically in FIG. **1** by communication paths **30**(*a*)–**30**(*c*) from the participants **22**(*a*)–**22**(*c*) to the certified trusted authority **26**.

At step **54** in FIG. **3**, the credential binding server **28** verifies the authenticity of the registration packets as being sent by the participants. The verification process itself is described in more detail with reference to FIG. **5**. At step **90**, the credential binding server **28** decrypts the symmetric bulk data encryption key K_{sym} using its own private key as follows:

$$D_{Kpri_binder}(K'_{sym})=K_{sym}$$

Once the symmetric key is recovered, the credential binding server **28** uses it to decrypt the participant's public keys and other data contained in the registration packet (step **92** in FIG. **5**). This can be shown as:

$$D_{Ksym}(\text{Data'\&}K')=\text{data}, K_{sign_pub_part}, K_{exc_pub_part}$$

Next, at step **94**, the credential binding server **28** decrypts the hash of the participant using the recovered participant's public signing key, $K_{sign_pub_part}$. This yields the hash, dS, as computed and concealed (encrypted) by the originating participant:

$$D_{Ksign_pub_part}(dS'_{part})=dS_{part}$$

The credential binding server **28** then performs a two-step verification technique to verify that the packet actually originated from the participant, and not an impostor. At step **96**, the credential binding server **28** recalculates the participant's digital signature by hashing the data contained in the decrypted registration packet using the same hashing function employed by the participant. The recalculated hash is then compared with the decrypted hash received as a digital signature, i.e., privately encrypted hash, in the registration packet (step **98** in FIG. **5**). If the two hashes match, the credential binding server is assured both that the registration packet was indeed signed by the participant and that the contents have not been subsequently altered.

Returning once again to FIG. **3**, after the credential binding server **28** verifies the registration packet as belonging to the participant, the next step **56** in the registration process is to generate a credential for the participant. The credential binding server **28** generates the credential. Each credential is unique to a particular participant, and will be used in future transactions as a means for identifying the

participants and for authenticating the participants to each other. The credential contains the participant's public signing key, public key exchange key, unique identifiers, validity dates, owner information, issuer information, and information about the participant determined in advance by owners and controllers of the particular commerce environment.

At step **58**, the credential binding server **28** attaches a digital signature of the trusted credential authority **26** to the credential. The digital signature is generated by encrypting a hash of the credential using the private key of the trusted credential authority **26**. The binder's digital signature will be used by the participants during transactions to verify that communications are between authorized participants who have properly registered with the trusted credential authority. At step **60**, the signed credential is transferred from the credential binding server **28** back over the communications system to the individual computing units **24**(*a*), **24**(*b*), and **24**(*c*) at respective participants **22**(*a*), **22**(*b*), and **22**(*c*). The is illustrated in FIG. **1** by the signed credentials **32**(*a*)–**32**(*c*) being returned to the computing units along communication paths **34**(*a*)–**34**(*c*).

Transaction Process

Following the registration process, the participants are ready to conduct their commercial activity. Unlike the registration process, however, the transaction process involves communication only among the participants to the transaction. There is no interaction between any of the participants and the trusted credential authority.

As shown in FIG. **2**, the commerce transaction process concerns the interchange of documents and instruments by participants, such as the representative three participants **22**(*a*)–**22**(*c*). The number of participants involved in a commerce transaction, the nature and content of the documents and instruments, and the exchange protocol are all defined in advance according to the particular commerce environment. Here, the participants **22**(*a*)–**22**(*c*) interchange information over the one or more communication systems between them, as illustrated by the pairs of arrows, without involving the credential authority **26**.

All commerce transactions have an originating participant, which is represented by participant **22**(*a*), and one or more recipient participants, which are represented by participants **22**(*b*) and **22**(*c*). The originating participant **22**(*a*) is the one who initiates a commerce transaction. The computing unit **24**(*a*) at the originating participant **22**(*a*) starts the commerce transaction by transmitting a request for the credentials of the intended recipient participants **22**(*b*) and **22**(*c*) who will be involved in the transaction (step **100** in FIG. **6**). The computing units **24**(*b*) and **24**(*c*) at respective recipient participants **22**(*b*) and **22**(*c*) return their unique credentials to the originating computing unit **24**(*a*) (step **102** in FIG. **6**).

At step **104** in FIG. **6**, the originating computing unit **24**(*a*) verifies the authenticity of the intended recipient participants. This verification is achieved by validating the digital signature of the trusted credential authority that is attached to the credential. Recall from the registration process described above that the credential binding server digitally signed each credential by encrypting the hash of the credential with the secret private signing key of the trusted credential authority. The originating computing unit **24**(*a*) decrypts the hash by decrypting the encrypted hash using the binder's public key. If the decrypted hash matches, bit-for-bit, an independently and locally (trusted) hash of the credential, then the originating computing unit is assured both that the credential was created and signed by the trusted credential authority and has not been subsequently modified, and that the recipient participant is thereby authenticated.

At step **106**, the originating participant **22**(*a*) constructs the appropriate commerce document and commerce instrument for the commercial transaction. A commercial transaction might involve more than one document or instrument, but for simplicity of discussion, our example transaction involves one document and one instrument. At step **108** in FIG. **6**, the originating computing unit **24**(*a*) computes the hash of the document and instrument through use of a hashing algorithm. At step **110**, the originating computing unit **24**(*a*) encrypts the hash using the private signing key of the originator's signing pair of asymmetric keys, as follows:

$$E_{K_{sign_pri_part}}(dS_{org_part})=dS'_{org_part}$$

Next, at step **112** in FIG. **7**, the originating computing unit **24**(*a*) generates symmetric bulk data encryption keys and encrypts the document and instrument using the symmetric keys. Preferably, the document is encrypted with one symmetric encryption key and the instrument is encrypted with a different symmetric encryption key. The encrypted hash, dS'_{org_part} attached to the document and instrument as a digital signature is also encrypted using the symmetric encryption keys. The bulk encryption step is represented as follows:

$$E_{K1_{sym}}(\text{document}+dS'_{org_part})=\text{document}'$$

$$E_{K2_{sym}}(\text{instrument}+dS'_{org_part})=\text{instrument}'.$$

At step **114** in FIG. **7**, the originating computing unit **24**(*a*) encrypts the symmetric keys with the public key exchange keys of the recipient participants that are intended to receive the document and instrument. For example, suppose the commerce document is a purchase order that is intended for the first recipient participant **22**(*b*), who is a merchant. Further, suppose the commerce instrument is a payment instruction that is intended for the second recipient participant **22**(*c*), which is a bank. The originating computing unit **24**(*a*) encrypts the first symmetric key used to encrypt the document (i.e., purchase order) and originator's digital signature with the first recipient's (i.e., merchant's) public key from its key exchange pair of asymmetric keys. This public key exchange key was obtained from the first recipient's credential. Similarly, the originating computing unit **24**(*a*) encrypts the second symmetric key used to encrypt the instrument (i.e., payment instruction) and originator's digital signature with the second recipient's (i.e., bank's) public key exchange key that was obtained from the first recipient's credential. Furthermore, sensitive or otherwise critical account information (such as a credit card account number) may be concatenated with the symmetric key, such that the account information also is encrypted with the recipient's (i.e., bank) public key exchange key. This affords some longer term protection for the account information, should the symmetric key or algorithm ever become compromised. This step is represented as follows:

$$E_{K_{keyexch_pub_1st_recip}}(K_{1sym})=K'_{1sym}$$

$$E_{K_{keyexch_pub_2nd_recip}}(K_{2sym})=K'_{2sym}$$

At step **116**, the originating computing unit **24**(*a*) transmits the signed encrypted document and instrument over the communication system to the computing unit **24**(*b*) at the first recipient participant **22**(*b*). This is illustrated diagrammatically in FIG. **2** by the document folder **36** and the instrument folder **38** being forwarded along communication path **40**. The originating computing unit **24**(*a*) also forwards the credential **32**(*a*) of the originating participant **22**(*a*) so

that the recipients will have the necessary information to return communication.

At step **118**, the computing unit **24**(*b*) at the first recipient participant **24**(*b*) decrypts the portion of the package that pertains to them. To continue our above example, the first recipient participant **24**(*b*) is a merchant and the commerce document is a purchase order that is intended for the merchant. Accordingly, the merchant computing unit **24**(*b*) decrypts the document portion. The computing unit **24**(*b*) initially decrypts the first symmetric bulk data encryption key using its own private key of the key exchange pair of asymmetric keys, as follows:

$$D_{K_{keyexch_pri_1st_recip}}(K'_{1sym})=K_{1sym}$$

The merchant computing unit **24**(*b*) then decrypts the document, or purchase order, using the decrypted symmetric bulk data encryption key, as follows:

$$D_{K1sym}(\text{document}')=\text{document}+dS'_{org_part}$$

This decryption also yields the originator's signature which itself is the hash of "document" encrypted with the signing private key of the originating participant. At step **120**, the first recipient computing unit **24**(*b*) verifies the originator's signature using the originator's signing public key which was provided in the originator's credentials that was forwarded with the document and instrument. If the decryption yields a hash that compares, bit-for-bit, with an independently, locally computed hash of "document", the first recipient participant **22**(*b*) can be assured that the originating participant **22**(*a*) sent the document, signed the document, and that the document was intended for the first recipient participant. The first recipient participant can authenticate the originating participant by verifying the digital signature of the credential binding authority on the signature credential of the first recipient in the manner described above (step **122** in FIG. **7**).

Notice that the method of this invention calls for the originating participant to forward both the document and instrument to the first recipient participant. In this manner, the commerce transaction is greatly simplified in that the originating recipient only forwards the items to one place. However, the first recipient participant can only decipher and read that portion of the package that is destined or intended for it. By encrypting those portions (in this case, the document) with the first recipient's public key exchange key, the originating participant is assured that only the first recipient can decrypt the document using its key exchange private key. Furthermore, by encrypting the other portions (in this case, the instrument) with the public key of the second recipient participant, the originating participant is assured that the first recipient participant cannot decrypt the instrument.

At step **124** in FIG. **8**, the computing unit **24**(*b*) at the first recipient participant **22**(*b*) repackages the encrypted instrument in another communications package and passes the package onto the computing unit **24**(*c*) at the second recipient participant **22**(*c*) over the same or a different communication system. This is illustrated diagrammatically in FIG. **2** by the instrument folder **38** being forwarded from the first recipient computing unit **24**(*b*) along communication path **42** to the second recipient computing unit **24**(*c*). The first recipient computing unit **24**(*a*) also sends along its credential **32**(*b*), and the originating participant's credential **32**(*a*), so that the second recipient participant will have the necessary information to return communication.

Steps **126**, **128**, and **130** are similar to steps **118**, **120**, and **122** that were described above. Briefly, the computing unit **24**(*c*) at the second recipient participant **24**(*c*) decrypts the commerce instrument (step **126**). In our example, the second recipient participant is a bank which decrypts the payment instructions concerning how the purchaser (i.e., the originating participant) intends to pay for the items ordered from the merchant (i.e., the first recipient participant). The instrument decryption involves a two-step process of (1) decrypting the second symmetric bulk data encryption key using its own private key exchange key, and subsequently (2) decrypting the instrument (e.g., payment instructions) using the recovered second symmetric key. This two-step procedure is represented as follows:

$$D_{K_{keyexch_pri_2nd_recip}}(K'_{2sym})=K_{2sym}$$

$$D_{K2sym}(\text{instrument}')=\text{instrument}+dS'_{org_part}$$

Notice that this decryption also yields the originator's signature, which itself is a hash of "instrument", encrypted with the private signing key of the originating participant. At steps **128** and **130**, the second recipient computing unit **24**(*c*) decrypts the originator's signature using the originator's public signing key that was provided in the originator's credentials and verifies the digital signature (by independently computing the hash of "instrument") and authenticity of the originating participant (by verifying the digital signature of the credential binding authority on the signature credential of the originating participant).

At step **132**, it is determined whether there are any more participants involved in the transaction. If there were more recipient participants (i.e., the "no" branch from step **132**), the second recipient would simply pass the remaining portions of the package, which would still be encrypted, onto the next participant (step **124**). The process of steps **126–130** would then be repeated. In our example, however, there are only two recipient participants: merchant and bank. Since there are no more participants (i.e., the "yes" branch from step **132**), the participants can return the appropriate receipts at step **134**.

If the transaction is approved by the second recipient, meaning that the bank determines that it can fulfill the originator's commerce instrument, the computing unit **24**(*c*) at the second recipient participant (i.e., the bank) returns a signed authorization receipt to the first recipient participant (i.e., the merchant) that payment is guaranteed. This is shown in FIG. **2** by the return receipt **44** being transmitted over communication path **46**. Once the first recipient computing unit **24**(*b*) receives the receipt, it will send a signed purchase receipt for the purchase to the originating computing unit **24**(*a*) and fill the purchase order. This is shown in FIG. **2** by the return receipt **48** being transmitted from the first recipient computing unit **24**(*b*) over communication path **49** to the originating computing unit **24**(*a*).

Alternate Cryptography Approaches

Although the preceding section provides a description of the process using public key cryptography, those skilled in the art can see that other cryptographic approaches exist.

In one embodiment, the symmetric keys used for the documents or instruments may be pre-established amongst the participants. This will reduce the overall canonical data size as well as eliminate several public key decryption operations, thereby enhancing performance. In such an embodiment, the keys may be exchanged or established through a variety of means.

In another embodiment, authentication of the participants, as well as the exchange of symmetric keys may be accomplished through a trusted third party. For example, the Kerberos system is one such architecture which facilitates this embodiment.

In yet another embodiment, the digital signature technique used may be based on a technology other than public key. For example, the El Gamal digital signature technique, based on symmetric key cryptography, may be used.

In yet another embodiment, the hash values from each of the document(s) and commerce instrument(s) might be concatenated and then encrypted using the originator's private signature key. This has the advantage of computing only one public key encryption for multiple documents.

Negotiated Off-line Encryption Architecture

In a number of countries, encryption is subject to stringent regulation. Limitations may be applied to the algorithms as well as the key sizes used. In order to enable and ensure compliance with applicable regulations, it may be necessary for the communicating participants to negotiate or otherwise pre-establish the applicable parameters. In transaction commerce electronically, the participants may not necessarily be in direct communication (e.g., they may be communicating via Electronic Mail).

In such a situation, it is necessary for the parties to have a trusted way of mutually agreeing on the appropriate encryption algorithms and key sizes. Given that the parties have already established a mutually trusted party (the certifying authority/binder), it follows that the binder can be trusted to provide an indicator for each participant on what that participant's limitations are. An index which indicates the strongest algorithm and key size is placed on each participant's credential.

When an originating participant encrypts a document or instrument for a specific recipient, that originating participant takes his or her encryption index, along with the encryption index of the intended recipient, and uses the two values to look up the appropriate algorithm in a preestablished table. This table typically is established by the certifying authority for the version of the commerce protocol being used.

The table typically contains the least common denominator of encryption allowed between the two participants. This is particularly useful in international commerce, where the regulations for one participant may differ from that of another. It is assumed (and the responsibility of the credential authority) that the algorithm/key size at the intersection in the table is one which is compliant with both countries involved in the transaction.

Table 1 illustrates an example preestablished table used in a negotiation process between a first participant with one possible encryption index and a second participant with another possible encryption index. In this example, the encryption indices range from a value "0" which represents the inability to encrypt items to a value "2" which represents the ability to use rather secure encryption techniques.

TABLE 1

	0	1	2
0	No Order Encryption	No Order Encryption	No Order Encryption
1	No Order Encryption	40 Bit RC4	32 Bit RC4
2	No Order Encryption	32 Bit RC4	64 Bit RC4

Communication Data Structure

FIG. **9** shows a data structure **140** used to carry each package that is exchanged between the participants, or between the participant and the credential trusted authority. It is used to encapsulate the messages and the message components. The data structure is a "tag-length-value" structure that provides a convenient way to handle fields as self-defined entities. This affords the flexibility of adapting to the message content being sent.

The tag-length-value (TLV) data structure **140** consists of three parts: an identifier field **142** (which is also known as the "tag"), a length field **144**, and a value field **146**. The identifier field or tag **142** is a fixed-size field (e.g., 32-bit) that defines or identifies the commerce data contained in the package. Only those tags defined for the particular commerce environment may be used. The length field **144** is a variable-sized field which contains a length of the commerce data contained in the package. The length field is preferably an exact byte count of the data contained in the value field **146**. For string fields that are NULL-terminated, the count includes the NULL character. The value field **146** is a variable-sized field which contains the actual commerce data defined by the tag.

There are many benefits resulting from this TLV data structure **140**. First, the data structure allows data elements to be self-describing, which affords tremendous flexibility. Second, it renders the data elements conducive to C++ programming protocols. Another benefit is that the TLV data structure facilitates future protocol extensibility. Still another reason is that the data structure enables backward compatibility. Further, the data structure permits customization of the contents of messages to fit the particular commerce environment without impacting the basic architecture, technology and implementation of the electronic commerce system itself.

Cryptography System Architecture

Each computing unit **24**(*a*)–**24**(*c*), as well as the credential server **28**, is equipped to perform cryptographic functions including encryption, decryption, digital signing, and verification. The computing units are programmed to execute a commerce application that facilitates the computerized, electronic commerce system. To sustain the security and authentication functions of the electronic commerce system, the commerce application must be able to provide encryption, decryption, and digital signing. Accordingly, each computing unit is implemented with a cryptography system that supports the commerce application with respect to these functions.

FIG. **10** shows a computing unit **22** that is used in the electronic commerce system of this invention. Computing unit **22** has a processor **150**, a network I/O port **152**, and a memory **154** which are all interconnected via an internal multi-bit bus **156**. The network I/O port **152** couples the computing unit **22** to the communication system employed in the electronic commerce system. For example, the network I/O port **152** might be in the form of a modem, network card, or the like. The computing unit **22** also includes a data input apparatus **158** (e.g., a keyboard, a mouse, a trackball, a keypad, etc.) and a card reader **160** (e.g., a smart card reader, a credit card reader, etc.) which are both operatively coupled to the bus **156**.

A commerce application **162** is stored in memory **154** and executable on the processor **150**. The commerce application is a software program that is tailored to the particular commerce activity in which the participant is involved.

The computing system **22** also has a cryptography system which supports the commerce application **162**. In the illustrated embodiment, the cryptography system is implemented in software that is stored in the memory **154** and executed on the processor **150**.

FIG. **11** shows the general architecture of the cryptography system, which is referenced with numeral **170**. Cryptography system **170** has three layers: (1) a cryptographic

application program interface (CAPI) **172**, which provides functionality to an application that it is supporting (in this case, the commerce application **162**); (2) one or more cryptographic service providers (CSP) **174**(*a*)–**174**(d), which implement the cryptographic functionality presented by CAPI to the application; and (3) one or more private application program interfaces (PAPI) **176**(*a*)–**176**(d) which allow the CSPs to communicate directly with a user **178**. As one example implementation, this cryptographic system could incorporated into an operating system as a service layer to provide cryptographic services, as is described below in more detail with respect to FIGS. **19**–**22**.

The CAPI layer **172** itself is thin. Its principal task is to select an appropriate CSP and verify its authenticity. When the commerce application **162** needs a sequence of cryptographic functions to be performed (e.g., encryption, decryption, signing), the application invokes the CAPI **172** to acquire a context associated with the appropriate CSP. The CAPI **172** then loads the CSP and verifies its authenticity. Each CSP is digitally signed by a certified trusted authority (such as the credential binding authority) through the use of its private encryption key. This digital signature **179** is graphically illustrated in FIG. **10** as part of the CSP **174** in memory **154**. The binding authority's public encryption key **180** is embedded in the CAPI **172** so that the CAPI **172** can verify the authenticity of the CSP by validating the digital signature **179** of the certified trusted authority. This verification prevents introduction of a foreign or impostor CSP. The CAPI **172** also provides an insulating layer between the application and the CSP so that the application never has direct access to the CSP, but can only call to the CSP through the CAPI.

The CAPI **172** is preferably implemented in software. As shown in the computing unit implementation of FIG. **10**, the CAPI **172** is stored in memory **154** and executed on processor **150**. The CAPI **172** is shown as having a CSP verifier module **181** to perform the above described CSP verification.

The CSPs **174**(*a*)–**174**(d) implement the cryptographic functionality requested by the application. In general, the CSPs perform encryption key management, encryption/decryption services, authentication and key exchanging tasks, hashing routines, and digital signing. Preferably, a different CSP is configured to perform each of these functions, although a single CSP can be implemented to perform them all. The CSPs **174**(*a*)–**174**(d) are dynamically accessible by the CAPI **172** using conventional loading techniques.

For the key management task, the CSP **174** has a key manager **182** (FIG. **10**) that stores, generates, or destroys encryption keys of any type, including symmetric cryptographic keys and asymmetric cryptographic keys. The CSP stores the participant's signing pair of private/public keys **184** that is used to digitally sign the registration packet, commerce document, and commerce instrument. The CSP also stores the participant's key exchange pair of private/public keys **186** that is used to decrypt messages sent by other participants.

The CSP **174** has a symmetric key generator **188** which generates the random symmetric bulk data encryption keys used to encrypt the messages sent to others. These symmetric keys are preferably "sessional" and thus generated for each transaction. The CSP also imports and exports encryption keys in their encrypted form. For example, the CSP can export the symmetric encryption keys in their encrypted form when the commerce document and the commerce instrument are dispatched to the recipient. Alternatively, the CSP can import such encrypted symmetric encryption keys from another participant. The CSP can also export or import the public encryption keys of the participants signing and key exchange pairs. These public keys are the only keys that are made readily available in their non-encrypted format. After importation, the CSP treats the encryption keys as if it generated them itself. Once the symmetric keys are used and the transaction is completed, however, the CSP destroys them.

The CSP **174** is specifically designed to avoid exposing the participant's private keys to any application or user. The encryption keys are never directly accessible to the applications. The asymmetric private encryption keys are not permitted to leave the CSP under any circumstances. In addition, the symmetric keys are permitted to leave the CSP only in an encrypted state; they are not exported in a raw form. In this manner, the CSP key manager prevents the application from ever inadvertently mishandling keys in a way that might cause them to be intercepted by those to whom they were not directly sent.

The CSP key manager assigns "handles" to the various keys that are created or imported. These handles are made available to the application via the CAPI. The application can manipulate the keys indirectly using the handles, rather than having absolute control over them. However, the keys themselves remain hidden from the application and CAPI.

One or more CSPs are also provided with encryption/decryption devices which encrypt and decrypt data using previously generated or imported symmetric keys, and the asymmetric key pairs. More particularly, the CSP **174** has a symmetric key encryption/decryption device **190** which is used to encrypt and decrypt messages (such as the commerce document and commerce instrument) using the generated or imported symmetric keys. An asymmetric key encryption/decryption device **192** is also provided to encrypt and decrypt the symmetric keys using the key exchange key pair **186**.

As mentioned above, another function of a CSP **174** is to perform an authentication/key exchange protocol. In this protocol, the CSP generates a symmetric key to encrypt the data and then encrypts that encryption key using the public signing key that belongs to the intended recipient. This protocol provides a high degree of security by encrypting the data, and then assuring that only the intended recipient can open the encrypted packet.

The CSP **174** also has a hashing calculator **194** which computes the cryptographic digest of the data contained in any messages sent between participants. The hashing calculator **194** translates the data according to a hashing function into a fixed-size, and often reduced, hash value which is representative of the original data. Cryptographic hash functions are beneficial in that they reduce the size of data to a hash value which is unique to that data set. Additionally, according to some cryptographic hash functions, it is computationally infeasible to find two data streams with the same hash value.

The CSP **174** also performs the task of digitally signing messages sent by the participant. The CSP has a signing device **196** which provides a digital signature unique to the combination of message and participant. Preferably, the CSP signing device **196** computes the digital signature by encrypting a previously computed hash value output by the hashing calculator **194** with the private signing key pair **184**.

The CSP **174** is preferably implemented in software as dynamic linked libraries (DLLs). This implementation is advantageous because it can be easily invoked by the CAPI or by the application through the CAPI. Furthermore, the

cryptographic functions can be changed or updated simply by replacing one or more DLLs. With the CAPI layer in between, the CSP DLLs can be replaced without affecting how the application interacts with them. Additionally, by packaging the cryptographic services in DLLs, it will be possible to change the strengths of the services as regulatory considerations change without impacting the higher level applications.

The CSPs might also be implemented in combination with hardware. One example is a set-top box implementation having a smart card reader and installed software. Here, the cryptographic keys are stored and used only in the card processor, so that they are not even exposed to sophisticated software attacks. Another example is a combination tamper-resistant hardware (such as a smart card) and a personal computer (PC). Here, the PC might not be as trusted as the set-top box, and thus extra verification of the hardware and terminal might be necessary.

Even with the protections enforced by the CAPI **172** and the CSPs **174**, there is a danger that a malicious application might attempt to request ostensibly legitimate operations that actually have the effect of compromising the user's security. For example, the application could have the CSP digitally sign a contract involving a large (and unwanted) financial commitment on the part of the user, unbeknownst to the user. Another example is for an application to instruct the CSP to export all symmetric session keys to the CSP of the application writer, giving the application writer the ability to read all of the user's encrypted data.

To guard against this threat, the CSPs have associated private application program interfaces (PAPI) **176**(*a*)–**176**(d) which permits the CSPs to interact with the user through an implementation-dependent user interface. Prior to operating on a particular transaction, the PAPI **176** presents an explanation of the transaction to the user to assist the user in understanding the specifics of the transaction. This presentation might be in the form of a textual summary displayed on a monitor. The PAPI enables the user to confirm or reject the transaction by either attaching a digital signature authorizing the transaction, or avoiding signing to cancel the transaction. By involving the user each time his/her signature is to be attached to a set of commerce documents and instruments, the cryptography system mitigates the danger that a malicious application will obtain sensitive information.

The PAPI **176** also provides several other functions. One function provided by the PAPI **176** is context verification in which the PAPI **176** verifies the authenticity of the user or entity that is operating at the computing unit prior to granting that user access to the application. For instance, the PAPI might mediate a user login sequence to validate the user. Alternatively, the PAPI **176** might be implemented in hardware, such as in combination with card reader **160** which reads a smart card or PCMCIA card, to verify the cardholder. The CSP **174** checks with the PAPI **176** to ascertain whether the appropriate authentication procedures have been met between the user and computing unit before letting the commerce application acquire a context. This context verification function is particularly useful for on-line PC services or banking ATMs that require participation from a person.

This aspect of the PAPI is implementation dependent. If implemented in a software-based PC, the context verification might simply involve a check with the PC operating system for the identity of the user already logged in. In a smart card terminal implementation, a full login with PIN entry may be required.

Another function of the PAPI **176** is to enable data entry from the user. The PAPI operates in conjunction with the data input apparatus **158** to permit data entry. Examples of this function include allowing the user to input a unique PIN during context verification, or to enter information regarding the transaction, or to enter confirmation/denial commands.

The PAPI **176** also informs the user as to the state of certain keys that the user might deem as requiring higher security. For example, the PAPI **176** can be configured to consult the user each time a key is created, exported, or used. This permits the user to have some control over particular keys, while further minimizing the possibility of a malicious application from exporting the keys in raw form.

The operation of the cryptography system **170** will now be described with reference to FIGS. **12–16**. Recall that each computing unit **24** is equipped with a cryptography system **170**, including an originating computing unit and a recipient computing unit. For purposes of discussion, the operation will be described in two parts. A first part explains the function of the cryptography system that takes place in the originating computing unit when preparing a message for transmission (FIGS. **12–14**). A second part describes the function of the cryptography system that occurs in the recipient computing unit after receiving a message (FIGS. **15–16**).

Part 1: Encrypting and Signing

At step **200** in FIG. **12**, the application **162** supplies a plaintext message to the CAPI **172** to be encrypted and signed. An example plaintext message is the commerce document and the commerce instrument. The CAPI **172** selects the appropriate CSP or CSPs **174** to perform the encryption and signing (step **202**). In one implementation, this step entails loading the appropriate DLL, and performing a series of calls, such as calls to begin and end the encryption and to digitally sign the result. For purposes of continuing discussion, the operation will be described as if the CSP **174** in FIG. **10** is capable of performing both the encryption and signing functions.

At step **204**, communication with the CSP is established between the CAPI **172** and selected CSP **174**. The CAPI **172** verifies the authenticity of the CSP **174** by validating the binding authority's digital signature **179** attached to the CSP **174** using the binding authority's public signature key **180** embedded in the CAPI **172** (step **206**).

Once the CSP is authenticated, the CAPI **172** passes the plaintext message to the CSP **174** for encryption (step **208** in FIG. **12**). The CSP hashing calculator **194** translates the plaintext message into a cryptographic digest (step **210** in FIG. **12**). During this time, an explanation of the transaction (such as what the commerce document or commerce instrument include) is passed to the PAPI **176** (step **212** in FIG. **12**). The PAPI **176** supports a user interface which presents the explanation to the user **178** (step **214** in FIG. **13**) and allows the user to input commands which confirm, modify, or deny the transaction. This affords an opportunity for the user to make changes to the transaction at its inception, before involving other participants. For example, the user might wish to change a purchase order or alter how he/she expects to pay for the purchased items. From a security perspective, this user interface ensures that the user is aware of the transaction and that he/she is authorizing it.

If the user authorizes the transaction, the PAPI **176** returns a confirmation to the CSP **174** (step **216** in FIG. **13**). At step **218**, the CSP signing device **196** attaches the originator's digital signature to the message by encrypting the cryptographic digest (hash) using the originator's private key of the signing pair. The CSP symmetric key generator **188**

generates a symmetric bulk data encryption key (step **220** in FIG. **13**) and the CSP symmetric key encryption/decryption device **190** encrypts the message and the originator's digital signature using the new symmetric encryption key (step **222**). At step **224**, the CSP asymmetric key encryption/decryption device **192** encrypts the symmetric encryption key using the key exchange public key of the intended recipient that is imported to the CSP.

At step **226** in FIG. **14**, the CSP **174** returns the signed and encrypted commerce document and commerce instrument to the CAPI **172**, and then onto the application. At the same time, the CSP **174** can inform the user via the PAPI **176** that certain keys, such as the symmetric key, are being exported from the CSP in its encrypted format (step **228** in FIG. **14**). However, the asymmetric private keys of the signing and key exchange pairs are not exported from the CSP, but are permanently retained in confidence within the CSP. The commerce instrument and commerce instrument are then ready for transmission from the originating computing unit to the recipient computing unit as described above.

Part 2: Decryption and Verification

The commerce application running at the recipient computing unit receives the signed encrypted document and instrument and passes the package to its own cryptography system **170**. Particularly, the encrypted document and instrument are supplied to the CAPI **172** for purposes of being decrypted and verified (step **250** in FIG. **15**). The CAPI **172** selects the appropriate CSP or CSPs **174** to perform the decryption and verification (step **252** in FIG. **15**). In this implementation, the appropriate CSP DLL is loaded and the application performs a series of calls to the DLL through the CAPI. For discussion purposes, the operation will again be described as if the CSP **174** in FIG. **10** is capable of performing both the decryption and authentication functions at the recipient computing unit.

Communication is then established between the CAPI **172** and selected CSP **174** (step **254** in FIG. **15**), and the CAPI **172** verifies the authenticity of the CSP **174** (step **256**). Once the CSP is authenticated, the CAPI **172** passes the encrypted document and instrument to the CSP **174** for decryption (step **258**). At step **260**, the CSP asymmetric key encryption/decryption device **192** decrypts the symmetric encryption key using the recipient's private key exchange key **186** maintained in the CSP. The CSP symmetric key encryption/decryption device **190** uses the recovered symmetric key to decrypt the message and originator's digital signature to provide the signed cryptographic digest (hash) (step **262** in FIG. **15**).

At this point, the CSP can verify the packet by decrypting the cryptographic digest (hash) using the originator's public signing key (step **264** in FIG. **16**). If the decryption yields a result that compares bit-for-bit with an independently, locally computed hash of the entire message, the participant is assured that the packet came from the originator and was not subsequently altered. For instance, in the electronic commerce system described above, the first recipient participant would be able to decrypt the commerce document, but decryption of the commerce instrument would yield meaningless information.

The plaintext message is returned from the CSP to the CAPI **172** and then to the commerce application **162** (step **266** in FIG. **16**). After the process is completed, the CSP destroys the symmetric encryption key for that session (step **268**).

EXAMPLE IMPLEMENTATIONS

The above discussion presents a general structure of an electronic commerce system. The following two cases provide example implementations of the electronic commerce system in specific commerce environments. The first example implementation is a credit card system and will be described with reference to FIGS. **17** and **18**. The second example implementation is an interactive television system and will be described with reference to FIG. **23**. In addition to these specific examples, the electronic commerce system can be implemented in a wide variety of commercial environments, including on-line services and debit or other banking card transactions.

Example 1: Credit Card System

FIGS. **17** and **18** diagrammatically illustrate an electronic credit card system **300** according to two different operation phases: a registration phase (FIG. **17**) and an order and purchase processing phase (FIG. **18**). The electronic credit card system **300** has several participants, including a purchaser **302**, a merchant **304**, an acquiring bank or acquirer **306**, an issuing bank **308**, and a trusted authority or binder **310**. The financial roles of each participant in the commerce transaction are well-known and will not be described herein.

Each participant is equipped with a computing unit, including a PC **312** at the purchaser and servers **314**, **316**, and **318** at the merchant, acquirer, and binder, respectively. Each computing unit is loaded with a credit card application and a cryptography system to support the credit card application with respect to its cryptographic needs.

During the registration phase of FIG. **17**, each participant requests and receives credentials from the binder **310**. "Binding" in this context means that the acquirer **306** is a known interface to an existing card payment system that is in use today; the merchant **304** is known to be able to accept credit cards for payment and deposit them with the acquirer **306**, and the purchaser **302** has a payment card known by the issuing bank **308** that issued the card. Credentials for the acquirer **306** and the merchant **304** are created by the binder **310** as authorized by their responsible bank. Credentials for the purchaser are created by the binder **310** as authorized by the purchaser's bank or authorized agent in the form of the card association.

Cardholder credentials are created when the purchaser **302** requests registration of a credit card. The purchaser enters card information (such as card number, expiration date, name on card) on the PC **312** to complete a registration application. The purchaser's computing unit also uses its cryptography system to generate the signing and key exchange pairs of public/private cryptography keys. The public keys are included in the registration packet.

The purchaser's computing unit digitally signs the card registration packet and sends the packet over a communication system to the binder's server **318**. The binder's server **318** validates the registration packet and creates a cardholder credential based on the information supplied by the purchaser. The cardholder credential includes the following information:

Cardholder Credential
1. Credential Serial Number.
2. Cardholder Name.
3. Hashed Account Number.
4. Cardholder's Signing Public Key.
5. Cardholder's Key Exchange Public Key.
6. Credential Expiration.
7. Version of Credential Format.
8. Encryption Index

The "credential serial number" is a number generated by the binder to uniquely identify the credential. The "hashed account number" is created using a one-way hashing algorithm, such as the well-known SHA (secure hash

algorithm), which easily creates a unique hash value representative of the account number while rendering the regeneration of the account number from the hash value computationally infeasible. The hash guarantees that the credential is tied to a specific credit card. However, knowledge of the credential does not give any insight to the actual number of the credit card. The "credential expiration" field contains the date range that the credential is valid as decided by the binder.

The cardholder credential is signed by either the binder **310**, which might be the card association binder, or the card issuing bank binder in those cases where the card association has authorized the card issuing bank to do so. This credential is then sent back over the communication path from the binder's server **318** to the purchaser's computing unit **312** as indicated by the credential **320** in FIG. **17**.

Notice that the acquirer **306** registers with the binder **310** for both itself and on behalf of the merchant **304**. In the credit card context, the merchant **304** participates based upon the authority of the merchant's financial institution, which is the acquirer **306** in this case. As a result, there is information known only to the acquirer that will be included in the merchant's credential. The binder will create both credentials **322** and **324** for the merchant and acquirer and send both to the acquirer, which then passes the merchant's credential **322** onto the merchant. Alternatively, if authorized, the acquirer can create the merchant's credential directly.

The merchant's credential is conceptually similar to the cardholder credential. There are eleven fields contained in the merchant's credential, with the first six fields being analogous to fields in the cardholder credential. The merchant's fields are:

Merchant Credential
1. Credential Serial Number.
2. Merchant Name.
3. Merchant's Signing Public Key
4. Merchant's Key Exchange Public Key
5. Period of Credential Validity.
6. Version of Credential Format.
7. Acquirer's Key Exchange Public Key (for payment instructions)
8. Card Brand Accepted
9. Acquirer BIN
10. Merchant ID
11. Encryption Index

The last four fields are used to help define the business relationships and responsibilities that are inherent in the credit card model. The "acquirer's key exchange public key" is included to enable encryption of the payment instruction by the purchaser such that only the acquirer may decrypt it. The acquirer's public is key could be distributed directly to the purchaser via the acquirer's credential, but is preferably included in the merchant's credential in the interest of efficiency and is in recognition of the special business relationship between the acquirer and the merchant.

The "card brand accepted" field indicates the particular card brand (e.g., Visa® or MasterCard®) that can be accepted by the merchant. The "acquirer BIN" is the identifier of the acquirer as used by the card association. The "merchant ID" is the identifier of the merchant as used by the acquirer. The cardholder's credential is signed by the card association binder or the acquirer if authorized by the binder.

The acquirer credential contains items similar to the purchaser or merchant credentials, including the following:

Acquirer Credential
1. Credential Serial Number.
2. Acquirer Name.
3. Acquirer's Signing Public Key
4. Acquirer's Key Exchange Public Key
5. Credential Expiration.
6. Version of Credential Format.
7. Acquirer BIN
8. Card Brand Accepted

The acquirer credential is digitally signed by the credit card association's credential binding authority.

Turning now to the order and purchase processing phase of FIG. **18**, the credit card system implemented with the electronic commerce system of this invention involves the communication paths shown in solid lines. The electronic credit card system of this invention facilitates the document and instrument interchange among the purchaser **302**, merchant **304**, and acquirer **306**. The dotted lines represent commerce processes implemented and operating by the existing payment card authorization and settlement systems in use today. As noted above, the purchaser PC **312**, merchant server **314**, and acquirer server **316** all execute a credit card order and payment application, and a cryptography system, to meet the security, privacy, integrity, and authenticity needs of this particular commerce system. During the order and processing phase, there is no active participation on the part of the binder **310**.

The purchaser **302** creates a commerce document in the form of a goods and services order (GSO) and a commerce instrument in the form of a purchase instruction (PI). The GSO and PI are configured in packets using the tag-length-value data structure described above with reference to FIG. **9**. An example GSO contains a name of the payee (merchant), an authorized amount, a unique transaction ID, purchaser's name and address, an order expiration, and an order form. An example PI includes the payee name (merchant), the authorized amount, the unique transaction ID, a GSO message digest, a card expiration, and order expiration.

The GSO and PI are digitally signed by the purchaser using the appropriate cryptographic service provider (CSP) in the cryptography system. The GSO and PI are then encrypted with a symmetric cipher using two different, randomly selected symmetric keys. The GSO data requires relatively less privacy (in comparison to the PI) because it contains less financial information. Thus, the GSO can be encrypted with a symmetric key of comparatively less strength, such as a key devised by the known RC4 algorithm with a key length of 40 bits. On the other hand, the PI contains more financial information and warrants higher privacy. The PI is encrypted with a DES symmetric key having a key length of 56 bits.

The symmetric keys for the GSO and PI are next encrypted with the key exchange public keys of the merchant and acquirer, respectively. The GSO symmetric key is less stringently protected by a relatively smaller 768-bit RSA public key, whereas the PI symmetric key is more strongly protected using a 1024-bit RSA key.

The signed encrypted GSO **330**, signed encrypted PI **332**, and the cardholder credential **320** are packaged and sent to the merchant over a communication network **334**. An example network **334** include a telephone network.

The merchant server **314**, using its cryptography system, decrypts the GSO **330** using its own key exchange private

key. The merchant server **314** then uses the purchaser's public signing key received in the cardholder credential **320** to verify the purchaser's digital signature on the GSO. This assures the merchant that the purchaser is authentic and the order is valid. Meanwhile, unable to decrypt the PI **332**, the merchant server **314** leaves the PI in its encrypted form.

To consummate the transaction, the merchant needs to secure funds from the purchaser to pay for the ordered items. Accordingly, the merchant server **314** sends the still encrypted PI **322** and the purchaser's credential **320** onto the acquirer server **316** using the same or another communication network **336**. An example network **336** might be an ISDN network which links the merchant and acquirer.

Using its cryptography system and own private key exchange key, the acquirer server **316** decrypts the PI **332**. The acquire server **316** then uses the purchaser's signing public key received in the purchaser's credential **320** to verify the purchaser's digital signature on the PI.

The PI is channeled through the existing payment card authorization system to validate the availability of funds for the purchaser. More particularly, the acquirer **306** uses the existing credit card network **338** to contact the issuing bank **308** that issued the credit card and handles that credit account for the purchaser **302**. If sufficient credit or funds are available, the issuing bank **308** returns an authorization response over the network **338** to the acquirer **306**. The authorization response is then digitally signed by the acquirer **306** and encrypted using the key exchange public key of the merchant. The response **340**, along with the acquirer's credential **324**, is sent over the communication network **336** to the merchant server **314**.

The merchant server **314** uses its cryptography system and private key exchange key to decrypt the response **340** and the acquirer's public signing key received in the acquirer's credential **324** to verify the response. Given the signed authorization, the merchant knows that payment is guaranteed. The merchant can now fill the order and ship the items to the purchaser. The merchant server **314** then generates a purchase receipt **342**, digitally signs the receipt, and encrypts it using the purchaser's public key exchange key. The purchase receipt **342** is sent from the merchant server **314** over the communication network **334** to the purchaser's PC **312** which then decrypts and verifies the receipt.

At the close of the merchant's business day, the merchant **304** requests payment for all approved payments for which goods have been shipped. The merchant server **314** transmits a file of all such payment transactions processed that day to the acquirer for decryption and deposit into the normal draft clearing and settlement system used today. The acquirer pays the merchant and then settles its account with the issuing bank using the existing credit card system **338**, and the issuing bank bills the purchaser in due course.

FIGS. **19–22** illustrate more particularly the use of the cryptography system as a lower-level service provider to an operating system application that is tailored for the purchaser (FIG. **19**), the merchant (FIG. **20**), the acquirer (FIG. **21**), and the binder (FIG. **22**). For convenience purposes, only the CAPI layer of the cryptography system is illustrated in these Figures.

As shown in FIG. **19**, a purchaser application **350** running at the purchaser PC **312** calls functions in a purchaser application program interface (API) **352** which then calls functions in the CAPI **172** of the cryptography system. The purchaser application **350** might request, for example, creation of a registration request, creation of a transaction, or processing of a transaction receipt. To create a registration request during the registration phase, the purchaser application **350** submits the binder credential and the purchaser's account number and the purchaser API returns a credential registration request. This request is properly signed and encrypted by the underlying cryptography system in the manner described above with respect to FIGS. **10–14**. As another example, to create a transaction, the purchaser application **350** passes a charge slip, a summary, the merchant credential, a transaction amount, purchase terms, and transaction details to the purchaser API **352**, which then returns a transaction and an identifier for that transaction.

Table 2 shows an example set of API function calls, listing input terms which are supplied from the purchaser application to the purchaser API and returned items sent back from the purchaser API.

TABLE 2

Purchaser API Function Calls

Function	Input Terms	Returned Terms
Create Credential Registration Request	Binder Credential Account Number	Request
Process Credential Registration Response	Registration Response	
Create Transaction	Charge Slip Summary Merchant Credential Transaction Amount Purchase Terms Transaction Details	Transaction Transaction Identifier
Process Transaction Receipt	Receipt	Amount Transaction Identifier Merchant Name Merchant Message

With respect to FIG. **20**, a merchant application **354** running at the merchant server **314** calls functions in a merchant API **356** which then calls functions in the CAPI **172** of the cryptography system. Table 3 shows an example set of function calls for the merchant API.

TABLE 3

Merchant API Function Calls

Function	Input Terms	Returned Terms
Create Credential Registration Request	Binder Credential Account Number	Request
Process Credential Registration Response	Registration Response	
Process Transaction	Transaction	Charge Slip Transaction Amount Purchase Terms Transaction Identifier Transaction Details Purchaser Credential
Create Transaction Authorization Request	Transaction Auth. Amount Transaction Identifier Charge Slip Acquirer Credential	Transaction Auth. Request
Process Transaction Authorization Response	Transaction Auth. Response	Transaction Auth. Identifier Authorized Amount Transaction Auth. Response Code Acquirer Message to Purchaser
Create Transaction	Authorized Amount	Receipt

TABLE 3-continued

Merchant API Function Calls		
Function	Input Terms	Returned Terms
Receipt	Transaction Identifier Merchant Name Merchant Message Purchaser Credential Acquirer Message to Purchaser	

With respect to FIG. 21, an acquirer application **360** running at the acquirer server **316** calls functions in an acquirer API **362** which then calls functions in the CAPI **172** of the cryptography system. Table 4 shows an example set of function calls for the acquirer API.

TABLE 4

Acquirer API Function Calls		
Function	Input Terms	Returned Terms
Create Credential Registration Request	Binder Credential Account Number	Request
Process Credential Registration Response	Registration Response	
Process Transaction Authorization Request	Transaction Authorization Request	Transaction Auth. Identifier Transaction Identifier Transaction Amount Transaction Auth. Amount Purchaser Name Purchaser Account Number Merchant Account Number Summary Purchaser Credential Merchant Credential
Create Transaction Authorization Response	Transaction Auth. Identifier Transaction Identifier Authorized Amount Purchaser Credential Merchant Credential Trans. Authorized Response Code	Transaction Auth. Response

With respect to FIG. 22, a binder application **364** running at the binder server **318** calls functions in a binder API **366** which then calls functions in the CAPI **172** of the cryptography system. Table 5 shows an example set of function calls for the binder API.

TABLE 5

Binder API Function Calls		
Function	Input Terms	Returned Terms
Process Credential Registration Request	Request	Requester Type Requester Name Requester Account Number Requester Signature Public Key Requester Key Exchange Pub. Key
Create Credential	Credential Type	Credential Regis.

TABLE 5-continued

Binder API Function Calls		
Function	Input Terms	Returned Terms
Registration Response	Requester Name Requester Account Number Binder Name Validity Period Requester Signature Public Key Requester Key Exchange Pub. Key Credential Identifier	Response Requester Credential

Example 2: Interactive Entertainment System

FIG. 23 diagrammatically illustrates an interactive television (ITV) system **400** that implements the electronic commerce system according to another aspect of this invention. The participants include a subscriber **402**, a merchant **404**, an acquire **406** and a cable operator **408**. The subscriber **402** is equipped with a computing unit in the form of a set-top box (STB) **412**. The merchant and acquirer each have a server **414** and **416**, respectively, and the cable operator **408** is equipped with a headend server **418**. Each computing unit is loaded with an ITV commerce application and a cryptography system to satisfy the security, privacy, integrity, and authenticity aspects of the ITV system. In this implementation, the ITV commerce application can be downloaded from the headend server to the STBs as requested, rather than remaining resident at the STB. The cryptography system, however, would reside and be executable at the STB.

The subscriber STB **412** and merchant server **414** are interconnected with the headend server **418** via an interactive network structure, which is represented by the network cloud **420**. One example implementation of the network structure is a hybrid fiber-optic/cable distribution system employing digital switching technologies such as asynchronous transfer mode (ATM) for bi-directional communications between the headend and individual subscriber/merchants. This multi-tier distribution system includes a high-speed, high-bandwidth fiber optic cable coupled between the headend and many regional distribution nodes. Each distribution node is then connected to multiple set-top boxes within the region via conventional home entry lines, such as twisted-pair telephone lines or coaxial cable. As an example, a single headend might service 250,000 or more subscribers, and each regional distribution node might support approximately 1200 subscribers. As technology continues to improve, parts of the ITV network structure can be replaced with wireless forms of communication, such as RF communication or satellite communication.

In this implementation, the cable operator **408** might perform several roles. The headend server **418** performs its traditional tasks of providing video content programs to the subscribers and facilitating communication over the network **420**. The cable operator **418** might also operate as the binding authority which certifies all subscribers connected to the ITV network, including the purchasing subscriber and the merchant subscriber. In this fashion, the subscribers submit registration applications over the ITV network **320**, and receive credentials from the headend server **418**. The cable operator **418** might also function as a financial institution analogous to an issuing bank in the credit card system, wherein the cable operator bills and collects money from the subscribers.

During a purchasing transaction, the purchaser subscriber **402** creates a commerce document in the form of an electronic purchase order (PO) **422** and a commerce instrument in the form of a purchase instruction (PI) **424**. The PO and PI are digitally signed by the purchaser subscriber STB, encrypted with selected symmetric keys that are then encrypted with the public key exchange keys of the merchant **404** and acquirer **406**, respectively. The signed encrypted PO **422**, signed encrypted PI **424**, and a subscriber credential **426** are packaged and sent over the interactive network **420** to the headend server **418**, which then redirects it to the merchant subscriber **404**.

The merchant **404** decrypts and verifies the PO **422**. The merchant **404** then sends the encrypted PI **424** and the purchaser subscriber's credential **426** onto the acquirer **406** via a separate communication network **428**. The acquirer server **416** decrypts and verifies the PI **424**. The acquirer determines whether the funds exist to consummate the purchase and, if so returns a signed authorization receipt **430** to the merchant. The merchant then fills the order and sends a signed receipt **432** to the subscriber.

Once the goods are shipped, the merchant **404** requests and receives payment for the goods from the acquirer. The acquirer then settles the account with the purchaser's issuing bank, or the cable operator **408** if authorized to perform that function.

The example embodiments illustrate several benefits of the electronic commerce system. First, each document or instrument is digitally signed and encrypted by an originating participant. This insures that the document and instrument will be decrypted and the signature independently verified only by the intended recipient participant. The recipient further verifies that the digital signature is valid for the document or instrument, and that the document or instrument was part of the initial package created by the originating participant and was not subsequently altered.

Another benefit is that the encrypted and signed packages are independent entities that can be transported by any communication protocol and system. This allows the electronic commerce system to operate over the communication system that already exists between the participants, such as phone networks, on-line services networks, wide area networks, interactive television networks, etc.

Another advantage is that the encryption attributes can be varied according to document type. In the credit card example, the GSO and PI are encrypted using independently specifiable cryptographic algorithms and strengths (often expressed in terms of key sizes or bit count). Regulatory and legal entities can therefore set the standards of allowable encryption for each document or instrument within a particular commerce system. The electronic commerce system according to an aspect of this invention can then be flexibly adapted to conform to those standards. In this manner, the electronic commerce system can be used in a wide variety of commerce activities, and support the particular regulatory and legal cryptographic standards imposed for each different activity. Furthermore, by implementing the cryptography system, and namely the cryptographic service providers (CSPs) as DLLs, the electronic commerce system can respond quickly to regulatory induced changes simply by replacing the DLLs without impacting the higher level application software.

The electronic commerce system has another benefit in that it can be used in a complementary fashion with existing commerce systems, such as the credit card payment card authorization and settlement system, that are already in place today.

In compliance with the statute, the invention has been described in language more or less specific as to structure and method features. It is to be understood, however, that the invention is not limited to the specific features described, since the means herein disclosed comprise exemplary forms of putting the invention into effect. The invention is, therefore, claimed in any of its forms or modifications within the proper scope of the appended claims appropriately interpreted in accordance with the doctrine of equivalents and other applicable judicial doctrines.

We claim:

1. A cryptography system to support an application requiring cryptographic functions, the cryptography system comprising:

a cryptographic application program interface (CAPI) to interface with the application and handle its requests for a cryptographic function;

at least one cryptography service provider (CSP) independent from, but dynamically accessible by, the CAPI;

the CSP providing the cryptographic function requested by the application, the CSP also managing and protecting at least one encryption key used in the cryptographic function to prevent exposure of the encryption key in a non-encrypted form to the CAPI and application; and

a private application program interface (PAPI) to interface the CSP with a user, the PAPI enabling the user to observe, confirm, or reject the requested cryptographic function.

2. A cryptography system as recited in claim **1** wherein the CSP is digitally signed with a digital signature of a certified trusted authority.

3. A cryptography system as recited in claim **2** wherein the CAPI verifies an authenticity of the CSP by validating the digital signature of the certified trusted authority.

4. A cryptography system as recited in claim **1** wherein the CSP computes a digital signature of the cryptography system.

5. A cryptography system as recited in claim **1** wherein the CSP stores at least one encryption key.

6. A cryptography system as recited in claim **1** wherein the CSP generates the encryption key.

7. A cryptography system as recited in claim **1** wherein the CSP destroys the encryption key following its use.

8. A cryptography system as recited in claim **1** wherein the CSP assigns a handle to the encryption key, the handle being made available to the application through the CAPI while the encryption key remains unavailable to the application and the CAPI.

9. A cryptography system as recited in claim **1** wherein, when the application requests encryption of a message, the CSP hashes at least some data contained in the message.

10. A cryptography system as recited in claim **1** wherein the CSP is implemented in software as dynamically linked libraries.

11. A cryptography system as recited in claim **1** wherein the CSP exports the encryption key only in an encrypted form.

12. A cryptography system as recited in claim **1** wherein, when the application requests encryption of a message, the CSP encrypts the message using a symmetric encryption key and then encrypts the symmetric encryption key using a public key from an asymmetric pair of private and public encryption keys.

13. A cryptography system as recited in claim **1** wherein the CSP manages at least one asymmetric pair of private and public encryption keys and permanently retains the asymmetric private encryption key in confidence.

14. A cryptography system as recited in claim 1 further comprising multiple CSPs to perform various ones of the cryptographic functions, said cryptographic functions including encryption, decryption, signing, and verification.

15. A cryptography system as recited in claim 1 wherein the PAPI presents, to the user, an explanation of the cryptographic function being requested by the application.

16. A cryptography system as recited in claim 1 wherein, when the application requests a digital signature on a message, the PAPI presents an opportunity for the user to confirm or deny attaching a digital signature before the CSP digitally signs the message.

17. A cryptography system as recited in claim 1 wherein the PAPI verifies the user's authenticity prior to enabling the user access to the application through the CSP.

18. A cryptography system as recited in claim 1 wherein the PAPI enables data entry from the user.

19. A cryptography system as recited in claim 1 wherein, when the application requests use or exportation of a particular encryption key, the PAPI selectively notifies the user when the particular encryption key is to be used or exported.

20. In a computer system having a processing unit and a computer-readable medium, a computer-implemented cryptography service provider stored on the computer-readable medium for execution on the processing unit as part of a cryptography system used to support a computer executable application requiring encryption or decryption of electronic messages to be sent or received by a user, the cryptography service provider comprising:

a key manager to manage encryption keys used to encrypt messages and to prevent the encryption keys from being exported in a non-encrypted form from the cryptography service provider;

an encryption/decryption device to encrypt or decrypt messages using the encryption keys; and

the cryptography service provider being configured as a dynamic linked library, software module which is dynamically accessible as needed by the application to receive a plaintext message and to return an encrypted message, or to receive an encrypted message and to return a plaintext message, without exposing the encryption keys in their non-encrypted form to the application.

21. A cryptography service provider as recited in claim 20 wherein the encryption/decryption device encrypts individual messages using a symmetric encryption key and then encrypts the symmetric encryption key using a public key from an asymmetric pair of private and public cryptographic keys.

22. A cryptography service provider as recited in claim 21 wherein the key manager exports the symmetric encryption key only in an encrypted form.

23. A cryptography service provider as recited in claim 20 wherein the key manager manages at least one asymmetric pair of private and public cryptographic keys and permanently retains the asymmetric private cryptographic key in confidence.

24. A cryptography service provider as recited in claim 20 where in the key manager stores at least one unique encryption key.

25. A cryptography service provider as recited in claim 20 further comprising a key generator to produce the encryption keys used to encrypt the message.

26. A cryptography service provider as recited in claim 20 wherein the key manager assigns handles to the encryption keys which are made available to the application, while maintaining the encryption keys themselves unavailable to the application.

27. A cryptography service provider as recited in claim 20 further comprising a hashing calculator to hash at least some data contained in the messages.

28. A method for supporting cryptographic functions requested by an application, the method comprising the following steps:

supplying a request for a cryptographic function to a cryptographic application program interface (CAPI);

selecting a cryptography service provider (CSP) to perform the desired cryptographic function;

establishing communication between the CAPI and the CSP;

verifying an authenticity of the CSP;

performing the cryptographic function at the CSP using at least one cryptographic key; and

preventing exposure of the encryption key in a non-encrypted form to the CAPI or application.

29. A method as recited in claim 28 wherein the performing step comprises performing a cryptographic function selected from a group comprising encryption, decryption, digital signing, and verification.

30. A method as recited in claim 28 further comprising presenting information concerning the requested cryptographic from the CSP through a private application program interface (PAPI) to a user to enable the user to observe, confirm, or reject the requested cryptographic function.

31. A method for encrypting a message comprising the following steps:

supplying a plaintext message to a cryptographic application program interface (CAPI);

selecting a cryptography service provider (CSP) for encrypting the message;

establishing communication between the CAPI and the CSP;

verifying an authenticity of the CSP;

passing the plaintext message from the CAPI to the CSP;

encrypting the message at the CSP using an encryption key maintained by the CSP to produce an encrypted message; and

passing the encrypted message from the CSP back to the CAPI without exposing the encryption key in its non-encrypted form.

32. A method as recited in claim 31 wherein the encrypting step comprises the following steps:

encrypting the message with a symmetric encryption key; and

encrypting the symmetric encryption key with a public key from an asymmetric pair of private and public cryptographic keys.

33. A method as recited in claim 32 wherein the encrypting step comprises the step of passing the encrypted symmetric encryption key to the CAPI along with the message.

34. A method as recited in claim 31 wherein the verifying step comprises the following steps:

attaching a digital signature of a certified trusted authority to the CSP; and

validating the digital signature to authenticate the CSP.

35. A method as recited in claim 31 further comprising the step of attaching a digital signature of the cryptography system to the message.

36. A method as recited in claim 31 further comprising the step of storing within the CSP at least one unique encryption key.

37. A method as recited in claim 31 further comprising the step of generating within the CSP the encryption key used to encrypt the message.

38. A method as recited in claim 31 further comprising the step of destroying the encryption key following its use.

39. A method as recited in claim 31 further comprising the following steps:

assigning a handle to the encryption key; and

making the handle available to the CAPI while maintaining the encryption key in confidence within the CSP.

40. A method as recited in claim 31 further comprising the step of hashing within the CSP at least some data contained in the message.

41. A method as recited in claim 31 further comprising the following steps:

passing an explanation of the message to a private application program interface (PAPI) used to interface the CSP with a user; and

presenting the explanation to the user.

42. A method as recited in claim 41 further comprising the step of verifying at the PAPI an authenticity of the user prior to presenting the explanation of the message.

43. A method as recited in claim 41 further comprising the step of enabling data entry from the user through the PAPI.

44. A method as recited in claim 41 further comprising the step of selectively notifying the user via the PAPI when a particular encryption key is to be used.

45. A computer-readable medium having computer-executable instructions for performing the steps in the method recited in claim 28.

46. A computer-readable medium having computer-executable instructions for performing the steps in the method recited in claim 31.

47. A computer-readable medium having a computer-executable instructions for implementing a cryptography system, comprising:

a cryptographic application program interface (CAPI) configured as a software module to interface with a computer-implemented application and to handle requests from the application for a cryptographic function;

at least one cryptography service provider (CSP) configured as a software module independent from, but dynamically accessible by, the CAPI;

the CSP providing the cryptographic function requested by the software application, the CSP also managing and protecting at least one encryption key used in the cryptographic function to prevent exposure of the encryption key in a non-encrypted form to the CAPI and software application.

48. A computer-readable medium as recited in claim 47, further comprising a private application program interface (PAPI) configured as a software module independent from the CAPI and CSP, the PAPI being configured to interface the CSP with a user and enable the user to observe, confirm, or reject the requested cryptographic function.

49. A computer-readable medium as recited in claim 47, wherein:

the CSP module is digitally signed with a digital signature of a certified trusted authority; and

the CAPI module verifies an authenticity of the CSP when accessing the CSP by validating the digital signature of the certified trusted authority.

* * * * *

UNITED STATES PATENT AND TRADEMARK OFFICE
CERTIFICATE OF CORRECTION

PATENT NO : 5,689,565
DATED : Nov. 18,1997
INVENTOR(S): Terence R. Spies et al.

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

On the title page, item [75] should read as follows:

-- [75] Inventors: Terence R. Spies, Redmond; Jeffrey F. Spelman, Duvall; Daniel R. Simon, Redmond, all of Wash.--

Signed and Sealed this

Twenty-seventh Day of April, 1999

Attest:

Attesting Officer

Q. TODD DICKINSON

Acting Commissioner of Patents and Trademarks